编程说明 版本03.2004

sinumerik

SINUMERIK 840D/840Di/810D
工作准备部分

SIEMENS

SIEMENS

SINUMERIK 840D/840Di/810D

编程说明－工作准备部分

编程说明

适用于

控制系统	*软件版本*
SINUMERIK 840D	7
SINUMERIK 840DE (出口版本)	7
SINUMERIK 840D powerline	7
SINUMERIK 840DE powerline	7
SINUMERIK 840Di	2
SINUMERIK 840DiE (出口版本)	2
SINUMERIK 810D	3
SINUMERIK 810DE (出口版本)	3
SINUMERIK 810D powerline	7
SINUMERIK 810DE powerline	7

版本 03.04

SINUMERIK® -文献

版本说明

以下是当前版本及以前各版本的简要说明。

每个版本的状态由“附注”栏中的代码指明。

在“附注”栏中的状态码分别表示：

A .... 新文件

B .... 没有改动，但以新的订货号重印

C .... 有改动，并重新发行

若某页的内容在上一个版本后有实质性的更改，则在该页的顶部用新版本号来指标。

版本	订货号	附注
03.04	6FC5298-7AB10-3RP0	C

注册商标

SIMATIC®, SIMATIC HMI®, SIMATIC NET®, SIROTEC®, SINUMERIK® 和 SIMODRIVE® 均为西门子公司的注册商标。本文件中的其他名称也可能是商标，任何第三者使用此商标将会侵犯注册商标所有人的权利。

其它信息可以上网查找：
www.siemens.com/motioncontrol

使用 WinWord V 9.0 和 Designer V 7.0 制作该文献。

控制系统有可能执行本文献中未描述的某些功能。但是这并不意味着在提供系统时必须带有这些功能，或者为其提供有关的维修服务。

本文献内容符合所描述的硬件和软件。但是可能会有一些差异，我们不能保证它们完全一致。文献中的有关信息会定期审核，而且一些必要的修改会包含在下一个版本中。欢迎提出改进建议。

保留技术变更权利。

订货号 6FC5298-7AB10-3RP0
在联邦德国印刷

西门子股份公司。

前言

资料结构

SINUMERIK资料分为3种类型：

- 一般文献
- 用户文献
- 制造商/维修文献

读者对象

该资料面向机床用户。该资料详细说明用户所必需的信息，从而可以方便地对SINUMERIK 840D/840Di/810D控制系统进行编程。

标准功能范畴

在该编程说明中描述了标准的功能范畴。机床制造商增添或者更改的功能，由机床制造商资料进行说明。

SINUMERIK 840D/840Di/810D
系统的其它资料的详细信息（比如说通用接口，测量循环…），您可以从当地西门子办事处获得。

控制系统有可能执行本文献中未描述的某些功能。但是这并不意味着在提供系统时必须带有这些功能，或者为其提供有关的维修服务。



适用性

该编程说明适用于控制系统：

SINUMERIK 840D	SW7
SINUMERIK 840DE (出口版本)	SW7
SINUMERIK 840D powerline	SW7
SINUMERIK 840DE powerline	SW7
SINUMERIK 840Di	SW2
SINUMERIK 840DiE (出口版本)	SW2
SINUMERIK 810D	SW3
SINUMERIK 810DE (出口版本)	SW3
SINUMERIK 810D powerline	SW7
SINUMERIK 810DE powerline	SW7

带操作面板 OP 010, OP 010C, OP 010S,
OP 12 或者 OP 15 (PCU 20 或者 PCU 50)

SINUMERIK 840D powerline

自2001年9月起：

- SINUMERIK 840D powerline 和
- SINUMERIK 840DE powerline

带有升级功能。**powerline** 中的模块列表参见硬件说明

- /PHD/ 章节 1.1

SINUMERIK 810D powerline

自2001年12月起：

- SINUMERIK 810D powerline 和
- SINUMERIK 810DE powerline

带有升级功能。**powerline**
中的模块列表参见硬件说明

- /PHC/ 章节 1.1

热线电话

有问题时请打以下热线电话：
A&D 技术支持 电话：+49 (0) 180 5050 – 222
传真： +49 (0) 180 5050 – 223
电子邮件： adsupport@siemens.com

资料方面有疑问时（建议，更正）请发传真或电子邮件至：
传真：+49 (0) 0131 98 – 2176
电子邮件： motioncontrol.docu@erlf.siemens.de
传真格式：参见手册封底所附带的表格。

英特网地址 www.siemens.com/motioncontrol

出口版本

在出口版本中不含有以下功能：

功能	810DE	840DE
5轴加工软件包	–	–
操作转换软件包（5轴）	–	–
多轴插补（>4轴）	–	–
螺旋线插补2D+6	–	–
同步动作，级别2	–	O[1)]
测量，级别2	–	O[1)]
适配控制	O[1)]	O[1)]
连续修整	O[1)]	O[1)]
使用编译循环（OEM）	–	–
垂度补偿，多维	O[1)]	O[1)]

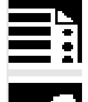
– 没有此功能
1) 有限的功能



资料编排结构

所有的循环和编程方法－只要可能－均按照相同的内部结构进行描述。通过划分为不同的级别，您可以很方便地找到所需要的信息。

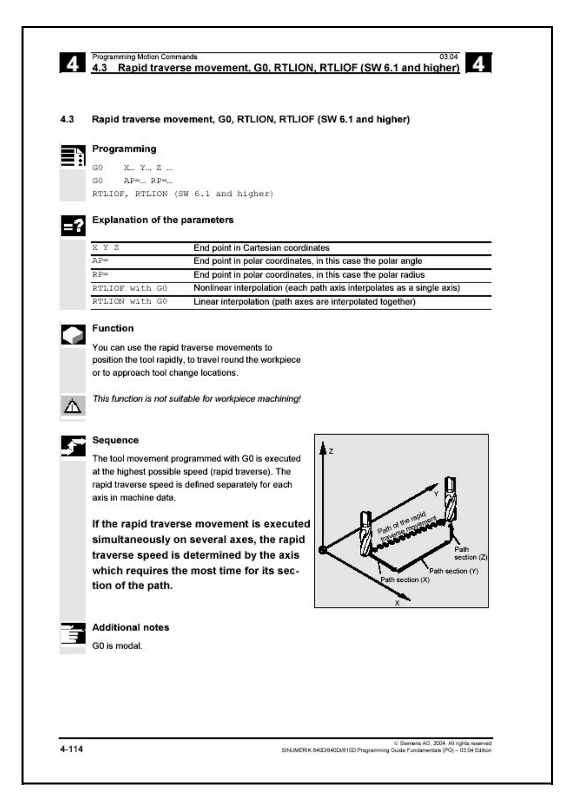
1. 快速一览



如果您要查看一个很少使用的指令，或者一个参数的含义，则您可以快速一览该功能如何编程，以及该指令和参数的说明。

这些信息总是位于一页的顶部。

说明：
由于篇幅的限制，对这样的指令和参数不可能说明所有可编程的方式。这里所介绍的编程方法仅仅是在车间现场最经常使用的一种。



4 Programming Motion Commands 03.04
4.3 Rapid traverse movement, G0, RTLION, RTLIOF (SW 6.1 and higher)

4.3 Rapid traverse movement, G0, RTLION, RTLIOF (SW 6.1 and higher)

Programming

```
G0    X… Y… Z …
G0    AP=… RP=…
RTLIOF, RTLION (SW 6.1 and higher)
```

Explanation of the parameters

X Y Z	End point in Cartesian coordinates
AP=	End point in polar coordinates, in this case the polar angle
RP=	End point in polar coordinates, in this case the polar radius
RTLIOF with G0	Nonlinear interpolation (each path axis interpolates as a single axis)
RTLION with G0	Linear interpolation (path axes are interpolated together)

Function

You can use the rapid traverse movements to position the tool rapidly, to travel round the workpiece or to approach tool change locations.

This function is not suitable for workpiece machining!

Sequence

The tool movement programmed with G0 is executed at the highest possible speed (rapid traverse). The rapid traverse speed is defined separately for each axis in machine data.

If the rapid traverse movement is executed simultaneously on several axes, the rapid traverse speed is determined by the axis which requires the most time for its section of the path.

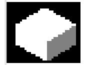
Additional notes

G0 is modal.

4-114
© Siemens AG, 2004. All rights reserved
SINUMERIK 840D/840Di/810D Programming Guide Fundamentals (PG) – 03.04 Edition

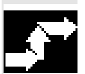
2. 详细说明

在理论部分，您可以找到下列问题的详细说明：



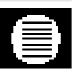
为什么需要该指令？

该指令有何作用？



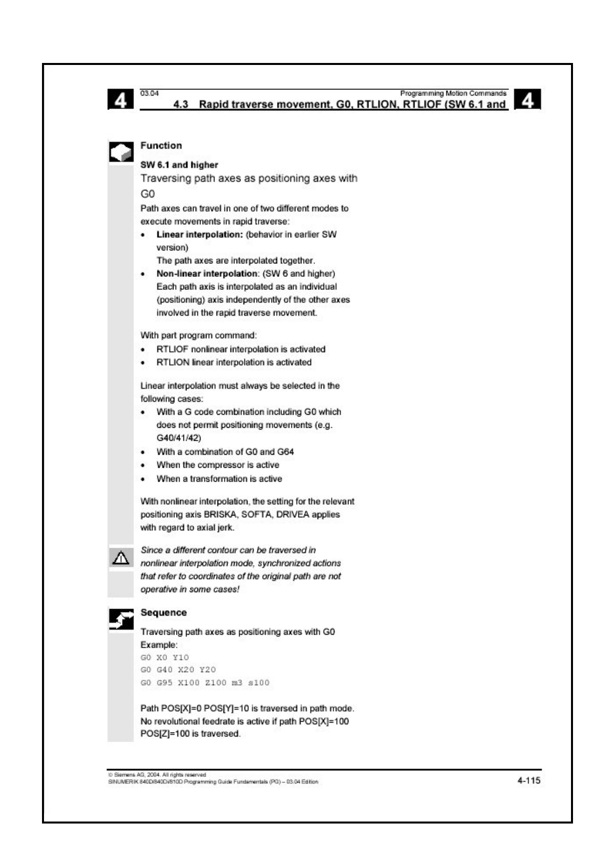
如何编程和执行？

这些参数有何作用？



还需要特别注意什么？

对于数控入门者来说，理论部分可以用作学习材料。至少要把此手册通读一遍，这样可以对SINUMERIK控制系统的功能范围和性能有一个初步的了解。



4 03.04 Programming Motion Commands 4

4.3 Rapid traverse movement, G0, RTLION, RTLIOF (SW 6.1 and

Function

SW 6.1 and higher

Traversing path axes as positioning axes with G0

Path axes can travel in one of two different modes to execute movements in rapid traverse:

- **Linear interpolation:** (behavior in earlier SW version)
  The path axes are interpolated together.
- **Non-linear interpolation**: (SW 6 and higher)
  Each path axis is interpolated as an individual (positioning) axis independently of the other axes involved in the rapid traverse movement.

With part program command:

- RTLIOF nonlinear interpolation is activated
- RTLION linear interpolation is activated

Linear interpolation must always be selected in the following cases:

- With a G code combination including G0 which does not permit positioning movements (e.g. G40/41/42)
- With a combination of G0 and G64
- When the compressor is active
- When a transformation is active

With nonlinear interpolation, the setting for the relevant positioning axis BRISKA, SOFTA, DRIVEA applies with regard to axial jerk.

Since a different contour can be traversed in nonlinear interpolation mode, synchronized actions that refer to coordinates of the original path are not operative in some cases!

Sequence

Traversing path axes as positioning axes with G0

Example:

```
G0 X0 Y10
G0 G40 X20 Y20
G0 G95 X100 Z100 m3 s100
```

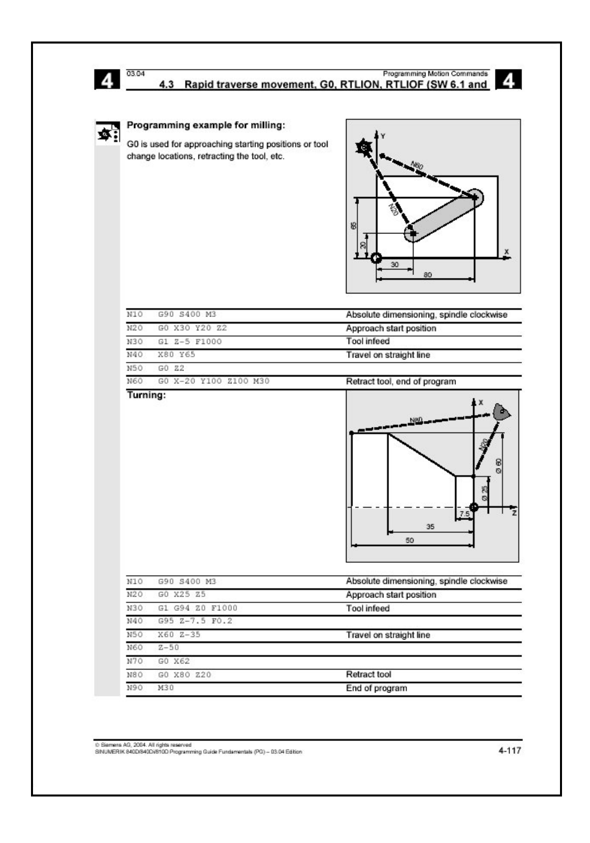
Path POS[X]=0 POS[Y]=10 is traversed in path mode.
No revolutional feedrate is active if path POS[X]=100 POS[Z]=100 is traversed.

© Siemens AG, 2004. All rights reserved
SINUMERIK 840D/840Di/810D Programming Guide Fundamentals (PG) – 03.04 Edition 4-115

3. 从理论到实践

通过编程示例可以了解如何在实践中使用这些指令。

在理论部分之后，所有的指令均有一个应用示例。



4 03.04 Programming Motion Commands 4

4.3 Rapid traverse movement, G0, RTLION, RTLIOF (SW 6.1 and

Programming example for milling:

G0 is used for approaching starting positions or tool change locations, retracting the tool, etc.

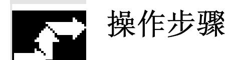
N10 G90 S400 M3	Absolute dimensioning, spindle clockwise
N20 G0 X30 Y20 Z2	Approach start position
N30 G1 Z-5 F1000	Tool infeed
N40 X80 Y65	Travel on straight line
N50 G0 Z2	
N60 G0 X-20 Y100 Z100 M30	Retract tool, end of program

Turning:

N10 G90 S400 M3	Absolute dimensioning, spindle clockwise
N20 G0 X25 Z5	Approach start position
N30 G1 G94 Z0 F1000	Tool infeed
N40 G95 Z-7.5 F0.2	
N50 X60 Z-35	Travel on straight line
N60 Z-50	
N70 G0 X62	
N80 G0 X80 Z20	Retract tool
N90 M30	End of program

© Siemens AG, 2004. All rights reserved
SINUMERIK 840D/840Di/810D Programming Guide Fundamentals (PG) – 03.04 Edition 4-117

符号说明


操作步骤


说明

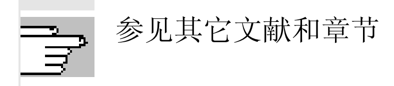

功能


参数


编程举例


编程


其它说明


参见其它文献和章节


注释和警告

原理

西门子的840D/840Di/810D是以先进的专业技术水平进行设计的，它们遵循相关的安全规范、标准和条例。

附加设备

西门子的控制系统可以在不同的应用场合，使用西门子提供的附加设备，进行扩展应用。

人员

只有相关的、受过培训的专业人员才允许使用该设备。没有受过培训的人员不可以操作系统，哪怕很短的时间。

必须明确地规定安装调试、操作及维护人员的职责，并且对他们的职责遵守情况进行监控和检查。

职能

在控制系统进行开机调试之前，必须要保证相关人员已经阅读并理解了该操作说明。此外，用户应该始终关注控制系统的总体技术状态（外部可以识别的故障、缺陷以及运行状态的改变）。

维修

维修工作只能由相关专业的、受过培训的合格人员进行，他们必须根据维修和维护手册的说明进行这些工作。在此，必须注意遵守相关的安全规范。

说明

以下行为被认为是不正确操作，因此生产厂家不承担责任：

与上面所述正确用法相违背的应用。

如果在非正常状态使用控制系统，或者不遵循安全规范、没有遵照使用说明中所作的操作要求而进行操作。

没有在系统的开机调试之前排除可能对安全造成隐患的故障。

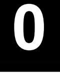
在控制系统中改变、跳转或者取消一些设备，它们有助于正常功能的使用和安全性能的发挥。



不正常的使用有可能造成不可预见的危险，它们会对：

- 人身安全造成危害，
- 也可能对系统、机床和企业与用户的其它财产造成损害；

在本资料中使用下面的说明表示特定的含义：



提示：

在资料中出现该符号时表示您需要对此性能引起重视。



当出现订货数据时，会有该符号。它提醒您系统只有匹配了该选件时才执行此功能。

警示符号

在本资料中使用下面不同的警示符号表示需要以不同的等级关注：



危险

该警示符号表示如果不采取相应的预防措施，将会造成严重的人身伤亡或者财产损失。



警告

该警示符号表示如果不采取相应的预防措施，则有可能会造成严重的人身伤亡或者财产损失。



小心

该警示符号（带三角符号）表明如果不采取相应的预防措施，则有可能引起轻微的伤害事故。

小心

该警示符号（不带三角符号）表明如果不采取相应的预防措施，则有可能引起财产损失。

注意

该警示符号表明如果不注意相应的提示，则可能会引起不好的结果或状态。

■

目录

特摆动令 5-189

框架 6-237

转换 7-269

灵活的NC编程 10-417

用于记录

灵活的NC编程

1.1 变量和计算参数



功能

通过使用变量而非固定值就可以灵活地编制程序。这样就可以对信号做出反应，比如测量值，或者通过使用变量而非固定值，可以把相同的程序用于不同的几何关系。

灵活的编程人员运用变量计算和程序转换可以建立一个高度灵活的程序档案，从而省去很多编程工作。



变量类型

控制系统区分出三种变量类型：

用户定义的变量	由用户定义的变量，带名称和类型，比如计算参数。
计算参数	专门的、预定义的计算变量，给定地址R及随后的数字。预定义的计算变量类型为REAL。
系统变量	供控制系统使用的变量，它们可以在程序中进行处理（写，读）。系统变量可以存取零点移位、刀具补偿、实值、轴的测量值、控制系统的状态等等（系统变量的意义参见附录）

变量类型

类型	*意义*	*值的范围*
INT	带符号的整数值	$\pm(2^{31} - 1)$
REAL	实数 (带小数点的分数, LONG REAL 按照 IEEE)	$\pm(10^{-300} \ldots 10^{+300})$
BOOL	逻辑值：TRUE (1) 与 FALSE (0)	1, 0
CHAR	1 个 ASCII 字符, 相应的代码	0 … 255
STRING	字符串, 字符数在 […]中, 最多 200 个字符	系列数值 带 0 … 255
AXIS	仅为轴名称（轴地址）	所有存在于通道中的轴名称 和主轴

FRAME	几何数据说明，用于偏移、旋转、比例、镜像，参见第 4 章。



计算变量

正常情况下，如果没有做进一步说明，则在地址R下有 100 个计算变量供使用，数据为实数型。



计算变量的具体个数（最大1000）由机床参数决定。

比如: R10=5

系统变量

控制系统提供系统变量，这些变量可以供所有程序运行并执行。

系统变量

提供机床版本和控制系统版本。它们有时不可以描述。



为了专门标识，系统变量的名称总是以 "$"
符号开始的。紧接着的是专门的名称。

系统变量类型一览

1. 字母	意义
$M	机床参数
$S	设定数据
$T	刀具管理参数
$P	程序数值
$A	实际数值
$V	服务参数

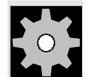
2. 字母	意义
N	NCK-全局
C	通道专用
A	轴专用

比如: $AA_IM
表示机床坐标系统中当前轴的实际值。

1.2 变量定义



用户定义变量
除了预设的变量，编程者还可以确定自己的变量，并用
数值加以注明。
局部变量（LUD）仅在其被定义的那个程序中才有效。
全局变量（GUD）在所有程序中都有效。
自软件版本 SW 4.4 起:
通过机床参数，在主程序中定义的局部用户变量（LUD）
可以重新定义为程序全局的用户变量（PUD）。



机床制造商

参见机床制造商说明。

当它们在主程序中被定义之后，它们在被调用的子程序的
所有级上都是有效的。它们随着零件程序起始而设置，随
着零件程序结束或复位而被删除。
举例:

```
$MN_LUD_EXTENDED_SCOPE=1

PROC MAIN          ;主程序
DEF INT VAR1       ;PUD-定义
...
SUB2               ;子程序调用
...
M30

PROC SUB2          ;子程序 SUB2
DEF INT VAR2       ;LUD-定义
...
IF (VAR1==1)       ;PUD 读
  VAR1=VAR1+1      ;PUD 读和写
VAR2=1             ;LUD 写
ENDIF
SUB3               ;子程序调用
...
M17
PROC SUB2          ;子程序SUB2
IF (VAR1==1)       ;PUD 读
  VAR1=VAR1+1      ;PUD 读取和
                   ;写
   VAR2=1          ;错误:LUD 自 SUB2
                   ;未知
ENDIF
...
M17
```



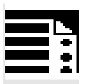
如果机床参数 $MN_LUD_EXTENDED_SCOPE
已经设定，就不能在主程序和子程序中用相同的名称再去定义一个变量。

变量名称

一个变量名称最多由31个符号组成。前面两个符号必须是字母或下划线。

符号 "$"

不能用于用户定义的变量，因为这个符号已经用于系统变量了。



编程

```
DEF INT 名称
或者 DEF INT 名称=数值

DEF REAL 名称
或者 DEF REAL 名称1,名称2=3,名称4
或者 DEF REAL 名称[数组索引1,数组索引2]

DEF BOOL 名称

DEF CHAR 名称
或者 DEF CHAR 名称[数组索引]=("A","B",…)

DEF STRING[字符串长度] 名称

DEF AXIS 名称
或者 DEF AXIS 名称[数组索引]

DEF FRAME 名称
```



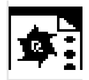
如果在定义时没有给变量赋值，那么系统将之预定为0。

变量必须在使用之前、在程序开始时定义。定义必须在一个独立的程序段中进行；每个程序段只能定义一个变量类型。



说明

INT	变量类型 整数型，意即整数的
REAL	变量类型实数，意即带小数点的分数
BOOL	变量类型布尔, 意即 1 或 0（TRUE 或者 FALSE）
CHAR	变量类型字符, 意即与 ASCII-代码相对应的字符 (0 到255)
STRING	变量类型字符串，意即符号串
AXIS	变量类型轴，意即轴地址和主轴
FRAME	变量类型框架，意即几何数据
名称	变量名称



编程举例

变量类型 INT

`DEF INT ANZAHL`	设定一个名为数目的整数型变量。 系统预设为 0。
`DEF INT ANZAHL=7`	设定一个名为数目的整数型变量。变量的起始值为 7。

变量类型 REAL

`DEF REAL TIEFE`	设定一个名为深度的实数型变量。 系统预设为 0。
`DEF REAL TIEFE=6.25`	会出现一个名为深度的实数型变量。变量的起始值为 6.25。
`DEF REAL TIEFE=3.1,LAENGE=2,ANZAHL`	在一行中可以确定多个变量。

变量类型布尔

`DEF BOOL WENN_ZUVIEL`	设定一个名为 WENN_ZUVIEL 的布尔型变量。 系统预设为 0（FALSE）。
`DEF BOOL WENN_ZUVIEL=1` 或者 `DEF BOOL WENN_ZUVIEL=TRUE` 或者 `DEF BOOL WENN_ZUVIEL`	设定一个名为 WENN_ZUVIEL 的布尔型变量。

变量类型字符

`DEF CHAR GUSTAV_1=65`	可以直接为字符类型的变量赋值一个相应的 ASCII-字符或
`DEF CHAR GUSTAV_1=65`	ASCII-字符的代码值(代码值 65 相应于字母 "A")。

变量类型字符串

`DEF STRING[6] MUSTER_1="ANFANG"`	字符串类型的变量可以接收一个字符链。字符个数的最大值列在变量类型后面的方括号中。

变量类型轴

`DEF AXIS ACHSNAME=(X1)`	轴类型的变量的名称为 ACHSNAME， 包含一个通道的轴名称—这里为 X1。（带有扩展地址的轴名称在圆括号中）

变量类型框架

DEF FRAME SCHRAEG_1	框架类型的变量名称为 SCHRAEG_1。

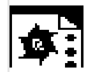


其它说明

轴类型的一个变量有一个通道的轴名称和主轴名称。

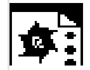
注意：

带扩展地址的轴名称必须写在圆括号中。



带程序局部变量的编程举例

DEF INT ZAEHLER	
SCHLEIFE: G0 X…	;循环
ZAEHLER=ZAEHLER+1	
IF ZAEHLER<50 GOTOB SCHLEIFE	
M30	



编程举例

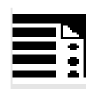
对当前几何轴的应答

DEF AXIS ABSZISSE;	;1. 几何轴
IF ISAXIS(1) == FALSE GOTOF WEITER	
ABSZISSE = $P_AXN1	
…	
WEITER:	

间接主轴编程

DEF AXIS SPINDLE	
SPINDLE=(S1)	
OVRA[SPINDLE]=80	;主轴倍率 = 80%
SPINDLE=(S3)	
…	

1.3 数组定义



编程

```
DEF CHAR NAME[n,m]
DEF INT NAME[n,m]
DEF REAL NAME[n,m]
DEF AXIS NAME[n,m]
DEF FRAME NAME[n,m]
DEF STRING[字符串长度] NAME[m]
DEF BOOL[n,m]
```

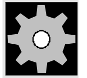
说明

INT NAME[n,m] REAL NAME[n,m]	变量类型 (CHAR, INTEGER, REAL, AXIS, FRAME, BOOL) n = 数组大小，第一个尺寸 m = 数组大小，第二个尺寸
DEF STRING[字符串长度] NAME[m]	字符串参数类型只能用一个尺寸的数组定义。
NAME	变量名称

字符类型的存储器规格对布尔型同样适用。

至软件版本 **SW3**:

数组的最大规格由机床参数设定。



机床制造商

参见机床制造商资料

类型	每个数组元的存储器占用量
BOOL	1 字节
CHAR	1 字节
INT	4 字节
REAL	8 字节
STRING	字符串长 + 1
FRAME	~ 400 字节, 取决于轴数量
AXIS	4 字节

最大的数组大小决定了存储块的规格，在此对变量存储器进行管理。此规格不应大于所需大小。

标准： 812 字节

如果没有定义大数组，

则选择： 256 字节.

自软件版本 SW4 起:

一个数组可能比一个存储块大。选择存储块规格的MD-值时应做到，
只在偶尔的情况下才出现数组的分块拼接。
标准：256 字节
每个单元的存储器占用量：同上
举例:
全局用户参数应该包含有关控制系统开与关的PLC机床参数。（用布尔数组定义）



其它说明

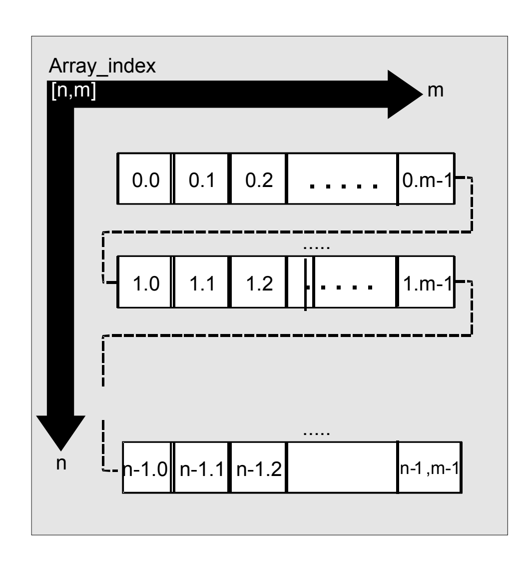
数组可以最多由 2 维尺寸定义。

带有字符串类型的变量的数组只允许是单尺寸。字符串长度根据字符串数据类型确定。

数组变址

可以通过数组变址读取数组单元。通过该数组变址,数组单元设置数值,或者读取数组单元值。

第一个数组单元从变址 [0,0] 开始；在数组大小为 [3,4] 时，最大可能的数组变址为 [2,3]。





在上面的例子中，数值在初始化时就已经被选定,这样它同时显示数组元的变址。由此可以显示各个数组单元的顺序。

数组的初始化

数组单元可以在程序运行时，也可以在数组定义时赋予初值。

在两维的数组中,右边的数组变址首先增值。

用数值表赋予初值, SET

1. 数组定义的方法

```
DEF Typ VARIABLE = 设定(值)
DEF Typ FELD[n,m] = 设定(值, 值, …)
```

或者

```
DEF Typ VARIABLE = 值
DEF Typ FELD[n,m] = (值`, 值 …)
```

- 有多少初值被编程就有多少数组元被赋值。
- 没有值的数组元(数值表中的空白)会自动被填上0。
- 轴类型的变量是不允许有空白的。
- 如果被编程的值超过现有的剩余数组元,就会触发系统警报。

举例:

```
DEF REAL FELD[2,3]=(10, 20, 30, 40)
```



在数组定义中您可以有选择的给出设置项。

2. 程序运行方法

```
FELD[n,m]= 设置(值, 值, 值,…)
FELD[n,m]= 设置(表达式, 表达式, 表达式,…)
```

- 赋予初值与定义时一样。
- 作为数值在此也可以为表达式。
- 赋予初值开始于数组变址编程。因此目标明确的部分数组也会被数值占用。

举例:

表达式语句

```
DEF INT FELD[5, 5]
FELD[0,0] = 设置(1, 2, 3, 4, 5)
FELD[2,3] = 设置(变量, 4*5.6)
```

在使用轴变量时,轴变址不能运行:

举例:

在一行里面赋初值

$MA_AX_VELO_LIMIT[1, AX1] = 设置(1.1, 2.2, 3.3)

与之相应的:

```
$MA_AX_VELO_LIMIT[1,AX1] = 1.1
$MA_AX_VELO_LIMIT[2,AX1] = 2.2
$MA_AX_VELO_LIMIT[3,AX1] = 3.3
```



用同样的数值赋初值, REP

1. 数组定义时的方法

```
DEF Typ FELD[n,m] = REP(值)
```

所有的数组元被相同的数值(常量)占用。



框架类型的变量不能被赋予初值。

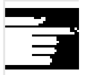
举例:

```
DEF REAL FELD5[10,3] = REP(9.9)
```

2. 程序运行方法

```
FELD[n,m] = REP(值)
FELD[n,m] = REP(表达式)
```

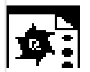
- 作为数值在此也可以使用表达式。
- 所有的数组元以相同的数值初始赋值。
- 赋予初值开始于编程的数组变址。因此目标明确的部分数组也可以被数值占用。



允许框架类型的变量,因此可以很方便地初始赋值。

举例:
用一个数值给所有的数组元赋初值。

```
DEF FRAME FRM[10]
FRM[5] = REP(CTRANS (X,5))
```



编程举例

初始化全部变量数组。
实际的数值占用情况见图。

```
N10 DEF REAL FELD1[10,3] = SET(0, 0, 0, 10, 11, 12, 20, 20, 20, 30, 30,
30, 40, 40, 40,)
N20 FELD1[0,0] = REP(100)
N30 FELD1[5,0] = REP(-100)
N40 FELD1[0,0] = SET(0, 1, 2, -10, -11, -12, -20, -20, -20, -30, , , ,
-40, -40, -50, -60, -70)
N50 FELD1[8,1] = SET(8.1, 8.2, 9.0, 9.1, 9.2)
```

Array index 2

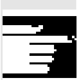
[1.2]	N10: Initialization with definition			N20/N30: Initialization with identical value			N40/N50: Initialization with different values		
	0	1	2	0	1	2	0	1	2
0	0	0	0	100	100	100	0	1	2
1	10	11	12	100	100	100	–10	–11	–12
2	20	20	20	100	100	100	–20	–20	–20
3	30	30	30	100	100	100	–30	0	0
4	40	40	40	100	100	100	0	–40	–40
5	0	0	0	–100	–100	–100	–50	–60	–70
6	0	0	0	–100	–100	–100	–100	–100	–100
7	0	0	0	–100	–100	–100	–100	–100	–100
8	0	0	0	–100	–100	–100	–100	8.1	8.2
9	0	0	0	–100	–100	–100	9.0	9.1	9.2
	The array elements [5.0] to [9.2] have been initialized with the default value (0.0).						The array elements [3.1] to [4.0] have been initialized with the default value (0.0). The array elements [6.0] to [8.0] have not been changed.		

1

1.4 间接编程



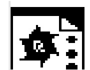
通过间接编程可以使程序通用。同时扩展地址(变址)
被合适类型的变量代替。



所有的地址都是可以设定参数的,除了：

- N － 程序段号
- G － G 指令
- L － 子程序

对于所有的可调整地址,间接编程是不可能的。
举例: X[1] 代替 X1 是不允许的。



编程

地址[变址]

编程举例

主轴

`S1=300`	直接编程
`DEF INT SPINU=1` `S[SPINU]=300`	间接编程： 转速 300 转/分钟的主轴，其编号被存储在 SPINU 变量中(在例一中)。

进给率

`FA[U]=300`	直接编程
`DEF AXIS AXVAR2=U` `FA[AXVAR2]=300`	间接编程： 定位轴进给,其地址名在轴类型变量中以变量名 AXVAR2 存储。

测量值

`$AA_MM[X]`	直接编程
`DEF AXIS AXVAR3=X` `$AA_MM[AXVAR3]`	间接编程： 轴机床坐标中的测量值,其名称被存储在变量 AXVAR3 中。

数组元

`DEF INT FELD1[4,5]`	直接编程

`DEFINE DIM1 AS 4`
`DEFINE DIM2 AS 5`
`DEF INT FELD[DIM1,DIM2]`
`FELD[DIM1-1,DIM2-1]=5`
	间接编程：
	在数组尺寸中，数组的大小必须作为固定值给出。

带轴变量的轴指令

`X1=100 X2=200`	直接编程

`DEF AXIS AXVAR1 AXVAR2`
`AXVAR1=(X1) AXVAR2=(X2)`
`AX[AXVAR1]=100 AX[AXVAR2]=200`
	间接编程：
	变量定义
	轴名称和轴运行，在变量中被存储为 100 和 200。

带轴变量的插补参数

`G2 X100 I20`	直接编程
`DEF AXIS AXVAR1=X` `G2 X100 IP[AXVAR1]=20`	间接编程： 定义和赋值轴名称 间接编程圆心

间接子程序调用

`CALL "L" << R10`	调用编号在R10中的程序



其它说明

R参数也可以被理解为带有简略的记数法的一维数组（R10 与 R[10] 相对应）。





间接 G 代码编程自软件版本 SW 5 起

G[<组-变址>] = <整数/实数−变量>

通过变量间接编程G代码，用于一个有效地循环编程



参数的意义

G<组-变址	整数常量，用以选出 G 代码组。
<整数/实数-变量>	整数或实数型变量，用以选出 G 代码号。



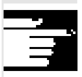
功能

间接 G 代码编程(自软件版本 SW 5 起)

通过变量间接编程G代码，可以进行有效的循环编程
。对此有两个参数

- G 代码组　整数常量
- G 代码号　整数/实数型变量

供使用。



有效的 G 代码组

只有模态有效的 G 代码组可以间接编程。
程序段有效的 G 代码组被拒绝，并给出警报
12470。

有效的 G 代码号

在 G 代码间接编程中不允许进行计算。
G代码号必须以整数或者实数型变量存储。无效的 G
代码号被拒绝，并发出报警 12475。必要的 G
代码号计算必须在独立的零件程序行中、在间接 G
代码编程前进行。

其它说明

所有有效的G代码都在该手册，章节“G功能/准备功
能清单”中被分成不同的组加以说明。详见
/PG/ 编程说明基础“列表”



编程举例

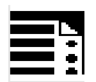
间接 G 代码编程

；可设定的零点偏移 G 代码组 8

N1010 DEF INT INT_VAR	
N1020 INT_VAR = 2	
...	
N1090 G[8] = INT_VAR G1 X0 Y0	；G54
N1100 INT_VAR = INT_VAR + 1	；G 代码计算
N1110 G[8] = INT_VAR G1 X0 Y0	；G55

；平面选择 G 代码组 6

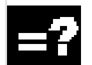
N2010 R10 = $P_GG[6]	；读当前平面的 G 代码
...	
N2090 G[6] = R10	；G17 – G19



字符串作为零件程序行处理

`EXECSTRING (<字符串-变量)`

通过指令EXECSTRING 一个零件程序行被间接处理



参数的意义

<字符串-变量>	字符串类型参数，用 EXECSTRING 传送。



功能

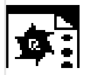
EXECSTRING (SW 6.4 以上)

通过零件程序指令 EXECSTRING
可以把一个字符串作为一个参数来传送，该字符串含
有一个需要执行的零件程序行。



其它说明

所有的零件程序结构都可以被取消，它们可以在零件
程序的程序部分被编程。PROC-和 DEF-
指令被排除在外，在 INI 和 DEF
文件中的应用也不行。



编程举例

间接零件程序行

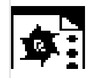
N100 DEF STRING[100] BLOCK	用于接收零件程序行的字符串变量
N110 DEF STRING[10] MFCT1 = "M7"	
N200 EXECSTRING(MFCT1 << " M4711")	执行零件程序行 “M7 M4711“
N300 R10 = 1	
N310 BLOCK = "M3"	
N320 IF(R10)	
N330 BLOCK = BLOCK << MFCT1	
N340 ENDIF	
N350 EXECSTRING(BLOCK)	执行零件程序行 “M7 M4711“

1.5 赋值

变量/计算参数可以在程序中被赋予一个合适类型的值。



赋值总是要求一个独立的程序段；每个程序段可以有多个赋值语句。给轴地址赋值（运行指令）与变量赋值相反要求一个分开的程序段。



编程举例

R1=10.518 R2=4 VARI1=45 X=47.11 Y=R2	一个数值的赋值
R1=R3 VARI1=R4	合适类型变量的赋值
R4=-R5 R7=-VARI8	用负号赋值（只有整数或实数类型时可以）

给字符串-变量赋值

在字符或字符串中区分大小写。

如果'或者"为字符链中的组成部分，则它们就被括 在'...' 中。

举例:

```
MSG ("Viene lavorata l'''ultima
figura")
```

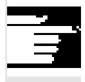
在显示屏上显示文本 'Viene lavorata l'ultima figura'。

不能显示的字符会以二进制或十六进制的常量接收到字符串中。

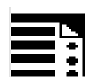
1.6 计算操作/计算功能



计算功能主要应用于 R 参数和实数型变量（或常量和功能）。整数型和字符型也是允许的。



在计算操作时，通常的数学计算有效。在处理中需优先处理的用圆括号给出。对于三角函数和它的反函数其单位是度(直角 = 90°）。



运算符/计算功能

+	加法
-	减法
*	乘法
/	除法 **注意** (Typ INT)**/**(Typ INT)**=**(Typ REAL); 比如: 3/4 = 0.75
DIV	除法,用于变量类型整数型和实数型 **注意** (Typ INT)DIV(Typ INT)**=**(Typ INT); 比如: 3 DIV 4 = 0
MOD	取模除法（整数型或者实数型），提供一个整数型除法的余数，比如 3 MOD 4=3
:	串运算符（在框架变量时）
Sin ()	正弦
COS()	余弦
TAN()	正切
ASIN()	反正弦
ACOS ()	反余弦
ATAN2(,)	反正切 2
SQRT()	平方根
ABS ()	总计
POT ()	2. 乘方（平方）
TRUNC ()	整数部分
ROUND ()	整数园整
LN ()	自然对数
EXP ()	指数函数
CTRANS ()	偏移
CROT ()	旋转

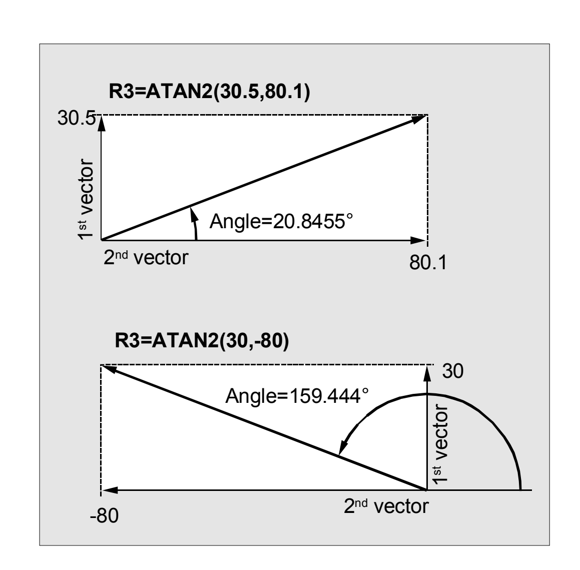
`CSCALE ()`	比例转换
`CMIRROR ()`	镜像

编程举例

`R1=R1+1`	新 R1 =旧 R1 +1
`R1=R2+R3  R4=R5-R6  R7=R8*R9`	
`R10=R11/R12  R13=SIN(25.3)`	
`R14=R1*R2+R3`	计算时先乘除后加减
`R14=(R1+R2)*R3`	括号先被计算
`R15=SQRT(POT(R1)+POT(R2))`	内括号首先被计算 R15 =($R1^2$+$R2^2$) 的平方根
`RESFRAME= FRAME1:FRAME2` `FRAME3=CTRANS(…):CROT(…)`	通过串计算操作，几个框架产生一个最终框架，或者框架的分量被赋值。

计算功能 ATAN2(,)

这个功能可以从两个互相垂直的矢量计算出总矢量的角度。计算结果位于四个象限之中(–180° < 0 < +180°)。角度是指在正方向的第二个数值。



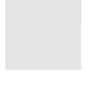
1.7 比较和逻辑计算操作

比较计算操作

比较运算可以用于类型为 CHAR, INT, REAL 和 BOOL 的变量。在字符型时代码值会被比较。

在数据类型为 STRING, AXIS 和 FRAME 时可以：== 和 <>.

比较计算操作的结果总是类型 BOOL。



比较操作可以用来表达转换条件。完整的表达式也可以进行比较。



比较运算符含义

==	等于
<>	不等
>	大于
<	小于
>=	大于等于
<=	小于等于

编程举例

```
IF R10>=100 GOTOF 目标
```
或
```
R11=R10>=100
IF R11 GOTOF 目标
```

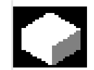
R10>=100 比较的结果首先存储在 R11 中。



比较错误时的精确度校正

```
TRUNC (R1*1000)
```

TRUNC 指令去除与精度系数相乘的操作数尾数。



功能

比较操作时的可设定精度

实数型零件程序参数内部用 64 位的 IEEE 格式描述。这种显示形式不能构成精确的十进制数，在与理想计算的数值进行比较时可能会带来不好的结果。

相对相等性

为了使这种描述所带来的不精确性不影响程序流程，在比较指令中不检测绝对奇偶性，而是检测一个相对相等性。

到软件版本 SW 6.3

所考虑的 10^{-12} 的相对相等性：

- 相同 (==)
- 不等于 (<>)
- 大于等于 (>=)
- 小于等于 (<=)
- 大于/小于 (><) 绝对相等

自软件版本 SW 6.4 起

所考虑的 10^{-12} 的相对相等性：

- 大于 (>)
- 小于 (<)



编程说明

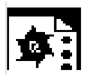
与实数型数据比较时，由于以上原因一般会出现一定的误差。如果不能接受所出现的偏差，则可以选择整数型计算，这时必须使运算数与一个精度系数相乘，然后用 TRUNC 截尾。

同步动作

所描述的比较指令性能也适用于同步动作。

兼容性

出于兼容性，通过设置 MD 10280:
PROG_FUNCTION_MASK Bit0 = 1，
可以取消对相对相等性(>)和(<)的检测。



编程举例

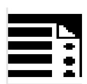
精度情况

N40 R1=61.01 R2=61.02 R3=0.01	赋予初值
N41 IF ABS(R2-R1) > R3 GOTOF FEHLER	执行跳转（到软件版本 SW 6.3）
N42 M30	程序结束
N43 FEHLER: SETAL(66000)	

R1=61.01 R2=61.02 R3=0.01	赋予初值
R11=TRUNC(R1*1000) R12=TRUNC(R2*1000) R13=TRUNC(R3*1000)	精确度校正
IF ABS(R12-R11) > R13 GOTOF FEHLER	没有执行跳转
M30	程序结束
FEHLER: SETAL(66000)	

构成并计算两个运算数的商

R1=61.01 R2=61.02 R3=0.01	赋予初值
IF ABS((R2-R1)/R3)-1) > 10EX-5 GOTOF FEHLER	没有执行跳转
M30	程序结束
FEHLER: SETAL(66000)	



逻辑运算

逻辑运算用于逻辑联系逻辑值。
AND, OR, NOT 和 XOR
只在布尔型变量时使用。通过隐含的类型转换它们也
可用于 CHAR，INT 和 REAL 数据类型。

在布尔的操作数和运算符之间必须加入空格。

在逻辑运算时（布尔运算），以下适用于数据类型
BOOL, CHAR, INT 和 REAL：
0 相当于 FALSE
非 0 相当于 TRUE



逻辑运算的意义

AND	与
OR	或
NOT	非
XOR	异一或

在算术表达中可以通过圆括号确定所有运算的顺序并
且由此脱离原来普通的优先计算规则。



编程举例

```
IF (R10<50) AND ($AA_IM[X]>=17.5) GOTOF 目标

IF NOT R10 GOTOB START
```

NOT 只与一个操作数有关。



位逻辑运算符

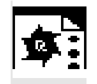
CHAR 和 INT

类型变量也可以进行位逻辑运算。如果有这种情况，
类型转换自动进行。



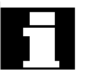
位逻辑运算符的意义

B_AND	位方式“与”
B_OR	位方式“或”
B_NOT	位方式“非”
B_XOR	位方式“异－或”



运算符 B_NOT

只与一个操作数有关；这个操作数在运算符之后。



编程举例

```
IF $MC_RESET_MODE_MASK B_AND 'B10000' GOTOF  ACT_PLANE
```

1.8 运算符的优先级



运算符的优先级

每个运算操作都被赋予一个优先级。在计算一个表达
式时，有高一级优先权的运算总是首先被执行。在优
先级相同的运算中，运算由左到右进行。

在算术表达式中可以通过圆括号确定所有运算的顺序
并且由此脱离原来普通的优先计算规则。

运算的顺序

从最高的优先级到最低的优先级

1.	NOT, B_NOT	非，位方式非
2.	*, /, DIV, MOD	乘，除
3.	+, -	加，减
4.	B_AND	位方式“与”
5.	B_XOR	位方式“异－或”
6.	B_OR	位方式“或”

7.	AND	与
8.	XOR	异－或
9.	OR	或
10.	<<	字符串的链接,结果类型字符串
11.	==, <>, >, <, >=, <=	比较运算符

IF语句的举例:

If (otto==10) and (anna==20) gotof end



级联运算符“：”在表达式中不能与其它的运算符同时出现。因此这种运算符不要求划分优先级。

1.9 可能的类型转换



赋值时的类型转换

常量的数值，变量或者给一个变量赋值的表达式必须和这个变量的类型兼容。一旦变量给出，在赋值时类型自动转换。

可能的类型转换

到 从	REAL	INT	BOOL	CHAR	STRING	AXIS	FRAME
REAL	是	是*	是[1]	是*	–	–	–
INT	是	是	是[1]	是[2]	–	–	–
BOOL	是	是	是	是	是	–	–
CHAR	是	是	是[1]	是	是	–	–
STRING	–	–	是[4]	是[3]	是	–	–
AXIS	–	–	–	–	–	是	–
FRAME	–	–	–	–	–	–	是

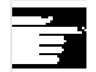
* 从实数型到整数型的转换中，小数值>= 0.5 时向上园整，否则舍去(ROUND 功能)。

[1] 值<> 0 对应于 TRUE, 值== 0 对应于 FALSE

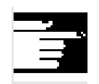
[2] 如果数值在允许的值范围内

[3] 如果只有一个字符

[4] 字符串长度 0 = >FALSE, 否则 TRUE



如果在转换中一个值大于目标范围，就会出现出错提示。



其它说明

如果表达式中出现混合类型，系统会自动进行类型匹配。

1.10 字符串运算

概述

除了在这一章描述的常规运算，如“赋值”和“比较”，还有其它的方法用于字符串的处理：

说明

类型转换到字符串：

STRING_ERG = <<bel._Typ[1]	结果类型： STRING
STRING_ERG = AXSTRING (AXIS)	结果类型： STRING

从字符串转换类型：

BOOL_ERG = ISNUMBER (STRING)	结果类型： BOOL
REAL_ERG = NUMBER (STRING)	结果类型： REAL
AXIS_ERG = AXNAME (STRING)	结果类型： AXIS

字符串的级联：

bel._Typ[1] << bel. Typ[1]	结果类型：STRING

大小写字母转换

STRING_ERG = TOUPPER (STRING)	结果类型：STRING
STRING_ERG = TOLOWER (STRING)	结果类型：STRING

字符串长度

INT_ERG = STRLEN (STRING)	结果类型：INT

在字符串中查找字符/字符串

INT_ERG = INDEX (STRING, CHAR)	结果类型：INT
INT_ERG = RINDEX (STRING, CHAR)	结果类型：INT
INT_ERG = MINDEX (STRING, STRING)	结果类型：INT
INT_ERG = MATCH (STRING, STRING)	结果类型：INT

部分字符串的选择

STRING_ERG = SUBSTR (STRING, INT)	结果类型：INT
STRING_ERG = SUBSTR (STRING, INT, INT)	结果类型：INT

单个字符的选择。

CHAR_ERG = STRINGVAR [IDX]	结果类型：CHAR
CHAR_ERG = STRINGFELD [IDX_FELD, IDX_CHAR]	结果类型：CHAR

[1] "bel._Typ" 用于变量类型 INT, REAL, CHAR, STRING 和 BOOL。

字符 0 的特别意义

在系统内部，字符 0
是被作为一个字符串的结束标识的。
如果一个字符被一个 0
字符代替，那么这个字符串就被缩短了。

举例:

`DEF STRING[20] STRG = "轴有."`	
`STRG[6] = "X"`	出现“X 轴停止”的提示
`MSG(STRG)`	
`STRG[6] = 0`	
`MSG(STRG)`	出现“轴”的提示

1.10.1 类型转换



因此不同类型的变量可以作为一个信息（MSG）的组成部分来使用。

到字符串的转换

使用运算符<<时，隐含适用于数据类型 INT，REAL，CHAR 和 BOOL （参见“字符串级联”）。
一个 INT
值会被转换为可读形式。在显示实数值时会给出小数点后 10 位。

AXIS 类型变量可以通过 AXSTRING 功能转换为 STRING。
FRAME 变量不能被转换。

举例:

```
MSG("位置:"<<$AA_IM[X])
```

字符串的类型转换

通过 NUMBER 功能可以实现从 STRING 到 REAL 的转换。

如果 ISNUMBER 提供值 FALSE, 则在调用 NUMBER 时，用相同的参数发出报警。

通过 AXNAME 功能，
可以实现一个字符串类型到轴类型的转换。如果该字符串没有分配到设计的轴名称，则给出报警。

句法

BOOL_ERG = ISNUMBER (STRING)	结果类型：BOOL
REAL_ERG = NUMBER (STRING)	结果类型：REAL
STRING_ERG = AXSTRING (AXIS)	结果类型：STRING
AXIS_ERG = AXNAME (STRING)	结果类型：AXIS

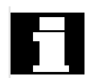
符号语义：

当字符串显示一个根据语言规则有效的实数时，
ISNUMBER （字符串）显示 TRUE。
由此可以检测这个字符串是否能被转换为一个有效的数字。
NUMBER（字符串）送回一个通过字符串显示的数，作为实数值。
AXSTRING（轴）作为字符串提供所给出的轴名称。
AXNAME（字符串）转换所给出的字符串为一个轴名称。

举例

DEF BOOL BOOL_ERG	
DEF REAL REAL_ERG	
DEF AXIS AXIS_ERG	
DEF STRING[32] STRING_ERG	
BOOL_ERG = ISNUMBER ("1234.9876Ex-7")	；现在：BOOL_ERG == TRUE
BOOL_ERG = ISNUMBER ("1234XYZ")	；现在：BOOL_ERG == FALSE
REAL_ERG = NUMBER ("1234.9876Ex-7")	；现在：REAL_ERG == 1234.9876Ex-7
STRING_ERG = AXSTRING(X)	；现在：STRING_ERG == "X"
AXIS_ERG = AXNAME("X")	；现在：AXIS_ERG == X

1.10.2 字符串的链接



这个功能使单个的字符串组合在一起。通过运算符实现级联：<<. 这个运算符适用于所有基本类型 CHAR，BOOL，INT，REAL 和 STRING 的组合，变成目标类型字符串。一个可能发生的必要的转换将根据现行的规则进行。FRAME 和 AXIS 类型不能使用这个算符。

句法：

bel._Typ << bel._Typ	结果类型：STRING

符号语义：

所给的字符串（有时隐含转换的其它类型）被相互级联在一起。

这个算符也可以作为所谓的“不变的”变量使用。这样可以执行一个明确的、到字符串类型的转换（FRAME 和 AXIS 不可用）。

句法

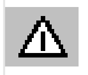
<< bel._Typ	结果类型：STRING

符号语义：

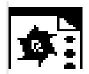
所给的类型被隐含转换为字符串类型。

比如，自文本列表组成一个信息或一个命令，并且插入参数（大约一个模块名）：

```
MSG(STRG_TAB[LOAD_IDX]<<BAUSTEIN_NAME)
```



在字符串级联时，中间结果不可以超过最大字符串长度。



编程举例

DEF INT IDX = 2	
DEF REAL VALUE = 9.654	
DEF STRING[20]STRG = "INDEX:2"	
IF STRG == "Index:" <<IDX GOTOF NO_MSG	
MSG ("Index:" <<IDX <<"/Wert:" <<VALUE)	；显示变址 2/值 9.654"
NO_MSG:	

1.10.3 字母大小写转换



这个功能允许一个字符串的所有字母以统一的标准显示。

句法

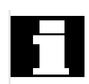
STRING_ERG = TOUPPER	(STRING)	结果类型：STRING
STRING_ERG = TOLOWER	(STRING)	结果类型：STRING

符号语义：
所有小写字母都变成大写或小写字母。

举例:
因为也可以由用户通过 MMC 输入，所以可以统一大小写字母：

```
DEF STRING [29] STRG
…
IF "LEARN.CNC" == TOUPPER (STRG) GOTOF LOAD_LEARN
```

1.10.4 字符串长度



这个功能允许确定字符串长度

句法：

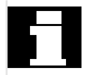
INT_ERG = STRLEN (STRING)	结果类型：INT

符号语义：
会有一系列字符被送回，这些数字从字符串开始计算都没有 0 字符。

举例:
与下面所说明的单个字符的存取相关，可以确定字符串结尾：

```
IF(STRLEN (BAUSTEIN_NAME) > 10) GOTOF FEHLER
```

1.10.5 在字符串中查找字符/字符串



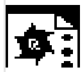
利用此功能，可以在后面一个字符串中查找单个字符或者一个字符串。查找结果说明：在字符串的一个位置找到需要查找的字符/字符串。

INT_ERG = INDEX	(STRING,CHAR)	结果类型：INT
INT_ERG = RINDEX	(STRING,CHAR)	结果类型：INT
INT_ERG = MINDEX	(STRING,STRING)	结果类型：INT
INT_ERG = MATCH	(STRING,STRING)	结果类型：INT

符号语义：
查找功能：它会把所查找字符串（第一个参数）中的位置送回。如果找不到字符或字符串，就会送回数值 -1。第一个字符位置为 0。

INDEX 在第一个参数中寻找作为第二个参数的字符（从前面）。
RINDEX 在第一个参数中寻找作为第二个参数的字符（从后面）。
MINDEX 与功能 INDEX 相对应，
除了传送一个字符表（作为字符串），从中第一个被找到的字符的变址被送回。
MATCH 在一个字符串中寻找一个字符串。

这样字符串可以按照一定的标准进行分解，大约是在
空格或路径分隔符的位置 ("/")。



编程举例

一个举例，说明如何分解输入路径名和模块名：

`DEF INT PFADIDX, PROGIDX`	
`DEF STRING[26] EINGABE`	
`DEF INT LISTIDX`	
`EINGABE = "/_N_MPF_DIR/_N_EXECUTE_MPF"`	
`LISTIDX = MINDEX (EINGABE, „M,N,O,P‟) + 1`	送回 3 作为 LISTIDX 中的值；因为“N”是前面的选择表中、参数 EINGABE 中的第一个字符。
`PFADIDX = INDEX (EINGABE, "/") +1`	由此 PFADIDX = 1有效
`PROGIDX = RINDEX (EINGABE, "/") +1`	由此 PROGIDX = 12有效
	通过在下一程序段引入的功能 SUBSTR，变量 EINGABE 可以被分成“路径”和“模块”。
`VARIABLE = SUBSTR (EINGABE, PFADIDX, PROGIDX-PFADIDX-1)`	然后传输 "_N_MPF_DIR"
`VARIABLE = SUBSTR (EINGABE, PROGIDX)`	然后传输 "_N_EXECUTE_MPF"

1.10.6 部分字符串的选择



这个功能允许一个部分字符串从一个字符串中生成。对此给出第一个字符的变址，有时还有所要求的长度。如果没有说明长度，则指的就是剩余字符串。

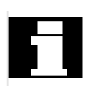
`STRING_ERG = SUBSTR    (STRING,INT)`	结果类型：INT
`STRING_ERG = SUBSTR    (STRING,INT, INT)`	结果类型：INT

符号语义：
在第一种情况，部分字符串从通过第一个参数确定的位置起到结束都被返还。
在第二种情况，结果字符串的长度被限制在通过第三个参数给出最大值之内。
如果开始位置位于字符串结尾之后，空字符串“ ”被返还。
如果开始位置或长度为非，触发警报。

比如:

`DEF STRING [29] ERG`	
`ERG = SUBSTR ("QUITTUNG: 10 bis 99", 10, 2)`	；由此 ERG == "10" 有效

1.10.7 单个字符的选择



这个功能允许选择一个字符串的单个字符。它不仅适用于正在读取的数据也适用于正在写入的数据。

句法：

`CHAR_ERG = STRINGVAR [IDX]`	结果类型：CHAR
`CHAR_ERG = STRINGFELD [IDX_FELD, IDX_CHAR]`	结果类型：CHAR

符号语义：

这个字符在这个字符串内被读取/写入，这个字符在给定的位置。如果给定的位置为非或大于这个字符串，触发警报。

信息举例：

在一个已经完成的字符串中使用一个轴标识。

```
DEF STRING [50] MELDUNG = "轴 n 已经到达位置"
MELDUNG [6] = "X"
MSG (MELDUNG)                          ；出现“X 轴到达位置”的信息
```

单个字符的存取只有在处理用户定义的变量（LUD-，GUD-和PUD-数据时才可能。
此外在调用一个子程序时只可以对“Call-By-Value”数据进行存取。

例如：

到系统数据、机床数据……的单个字符的数据存取：

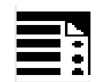
`DEF STRING [50] STRG`	
`DEF CHAR QUITTUNG`	
…	
`STRG = $P_MMCA`	
`QUITTUNG = STRG [0]`	；应答部件运用

Call-By-Reference-参数处理时的单个字符存取：

`DEF STRING [50] STRG`	
`DEF CHAR CHR1`	
`EXTERN UP_CALL (VAR CHAR1)`	；Call-By-Reference-参数!
…	
`CHR = STRG [5]`	
`UP_CALL (CHR1)`	；Call-By-Reference
`STRG [5] = CHR1`	



1.11 CASE 指令



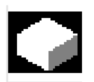
编程

```
CASE (表达式) OF 常量 1 GOTOF LABEL1 … DEFAULT GOTOF LABELn
CASE (表达式) OF 常量 1 GOTOF LABEL1 … DEFAULT GOTOF LABELn
```



命令解释

CASE	跳转指令的关键词
GOTOB	以反向跳转为目标的跳转指令(方向为程序开头)
GOTOF	以正向跳转为目标的跳转指令(方向为程序结尾)
GOTO	跳转目标首先是正向然后是反向的跳转指令(方向先是程序结尾然后是程序开头)
GOTOC	删除警报 14080 “跳转目标没有找到。 跳转目标首先是正向然后是反向的跳转指令(方向先是程序结尾然后是程序开头)
LABEL	目标(在程序内标记)
LABEL:	在跳转目标名称后有一个冒号。
Ausdruck	数学表达式
Konstante	INT 类型常量
DEFAULT	程序路径;如果没有前面的常量相吻合



功能

这个CASE 指令提供根据 INT
类型实际值不同而进行转移的可能。



运行

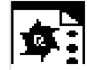
被 CASE
指令检测的常量采用什么值,程序就转移到所属跳转目标
确定的位置。



关于 GOTO
命令的更多信息详见第十章计算参数和程序跳转
对于常量没有采用前面确定的值的情况,可以用
DEFAULT 指令确定跳转目标。



如果 DEFAULT
指令没有被编程,在这些情况中紧跟在 CASE
指令之后程序段将成为跳转目标。



编程举例

举例 1

CASE(表达式) OF 1 GOTOF LABEL1 2 GOTOF LABEL2 … DEFAULT GOTOF LABELn
“1“ 和 “2“ 是可能的常量。 当表达式的值 = 1 (INT-常量)，跳转到带 LABEL1 的程序段 当表达式值 = 2 (INT-常量),跳转到带 LABEL2 的程序段 … 依此类推跳转到带 LABELn 的程序段

举例 2

```
DEF INT VAR1 VAR2 VAR3
CASE(VAR1+VAR2-VAR3) OF 7 GOTOF MARKE1 9 GOTOF MARKE2 DEFAULT GOTOF MARKE3
MARKE1: G0 X1 Y1
MARKE2: G0 X2 Y2
MARKE3: G0 X3 Y3
```

1.12 控制结构

解释

IF-ELSE-ENDIF	二选一
LOOP-ENDLOOP	无限循环
FOR-ENDFOR	计数循环
WHILE-ENDWHILE	在循环开头有条件的循环
REPEAT-UNTIL	在循环结尾有条件的循环



功能

控制系统按照编制好的标准顺序处理NC程序段。

用这些命令除了能确定在这一章描述的程序跳转,还能确定二选一和程序循环。

这些命令使编程具有某种结构并使程序具有较强的可读性。



运行

1. IF–ELSE–ENDIF

IF–ELSE–ENDIF-模块用于二选一：:

IF (表达式)
NC 程序段
ELSE
NC 程序段
ENDIF

如果表达式值为 TRUE,
也就是说条件被满足,这样后面的程序模块被执行。如果条件不满足, ELSE 分支被执行。
这个 ELSE 分支可取消。

2. 无限程序循环 LOOP

无限循环在无限程序中被应用。在循环结尾总是跳转到循环开头重新进行。

LOOP
NC-程序段
ENDLOOP

3. 计数循环 FOR

当一个带有一个确定值的操作程序被循环重复，FOR循环就会被运行。记数变量同时会从初始值到最后值增加数值初始值必须小于最后值。变量必须属于INT类型。

FOR 变量 = 初始值 **TO** 最后值
NC 程序段
ENDFOR

4. 在循环开头带有条件的程序循环 WHILE

只要条件满足，WHILE 循环就被执行。

WHILE 表达式
NC 程序段
ENDWHILE

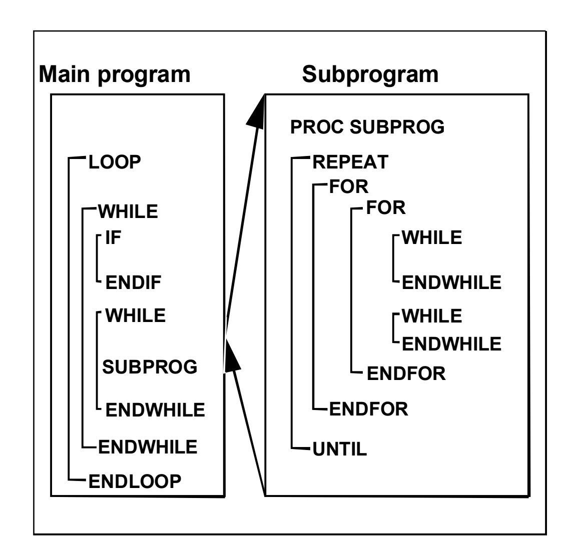
5. 在循环结尾带有条件的程序循环 REPEAT

REPEAT循环一旦被执行会不断重复,直到条件被满足为止。

REPEAT
NC程序段
UNTIL (表达式)

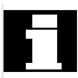
嵌套的层数

控制结构对部分程序有效。在每个子程序之内,嵌套的层数可以达到 8 个标准控制结构。



操作时间的实现

在标准有效的翻译操作中,可以通过程序跳转的运用达到比标准控制结构快的程序操作。
在前面汇编的循环中,程序跳转和标准控制结构没有实际的区别。



界限条件

带有标准控制结构数组元的程序段不能被跳过。在这些程序段中不允许有标签。

标准控制结构被翻译。在识别一个循环结尾时,考虑到所找到的标准控制结构，会寻找循环开头。
之后在翻译过程中,模块结构不会完全被检测。
建议不要混合使用标准控制结构和程序跳转。
在循环的预处理中,会检查控制结构的正确嵌套。



控制结构只有在一个程序的指令部分才可能。程序头的定义不能有条件或重复执行。

标准控制结构的关键词和跳转目标一样不能和宏叠加。
宏定义时不能进行检测



编程举例

1. 无限程序

```
%_N_LOOP_MPF
LOOP
      IF NOT $P_SEARCH                        ;没有程序段寻找操作
            G01 G90 X0 Z10 F1000
            WHILE $AA_IM[X] <= 100
                  G1 G91 X10 F500             ;钻孔图
                  Z-5 F100
                  Z5
            ENDWHILE
            Z10
      ELSE                                    ;程序段寻找过程
MSG("Im Suchlauf wird nicht gebohrt")
```

```
      ENDIF
      $A_OUT[1] = 1                          ;下一个钻孔板
      G4 F2
ENDLOOP
M30
```

2. 确定的零件数的生产

```
%_N_STUECKZAHL_MPF

DEF INT STUECKZAHL
FOR STUECKZAHL = 0 TO 100
      G01 …
ENDFOR
M30
```

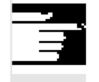
1.13 程序协调

通道

一个通道可以独立地处理自己的程序，而与其它的通道无关。这样，那些它所赋值的轴和主轴可以通过程序控制。
在安装调试时，控制系统可以设定两个或多个通道。

程序协调

如果多个通道和一个工件的生成有关,那么就要求同步程序操作过程。
对于程序协调有特殊的指令(命令)。它们总是在自己的程序段内。



说明
自软件版本 SW 5.3 起，
可以在自身的通道中进行程序协调。

程序协调的指令

- 给出绝对路径

```
INIT )(n,"/_HUGO_DIR/_N_name_MPF" )

INIT (n,"/_N_MPF_DIR/_N_name_MPF" )
```

在这里绝对路径根据以下规则构成

- *aktuelles Directory/*_N_*name*_MPF
  "实际目录"是指所选的工件目录或标准目录 */_N_MPF_DIR*。
- 在一定的通道中选择一个确定的程序来执行:
  n: 通道的序号,根据控制系统配置得到的值
- 完整的程序名

版本 SW 3 以前

在一个 **init**命令 (没有同步)和一个 **NC-Start** 之间必须至少有一个可执行的程序段。

在调入子程序时必须在路径中加上 "_SPF"。

比如:

```
INIT(2,"/_N_WKS_DIR/_ABRICHT_MPF")
G01 F0.1
START

INIT (2,"/_N_WKS_DIR/_N_UNTER_1_SPF")
```

- 给出相对路径

比如:

```
INIT(2,"ABRICHT")
```

在给出相对路径时，子程序调用的规则同样有效。

```
INIT(3,"UNTER_1_SPF")
```

在调用子程序时必须在程序名中加上 "_SPF"。

`START (n,n)`

在其它的通道中启动所选的程序。

n,n: 列举通道号:根据每个操作结构得出的值

`WAITM (标记号,n,n,...)`

在自身的通道中设定标记"标记号"。以准停结束上述的程序段。等待在给出的通道 "n" 中用相同的"标记号"标注的标记(自身通道不需要做说明)。标记会在同步之后被取消。

最多可以同时有 10 个通道被标记。

WAITMC (标记号,n,n…)	在自身的通道里设定标记“标记号“。准停只有在其他通道没有达到标记时才会进行。等待在给出的通道 ”n” 中用相同的”标记号”标注的标记(自身通道不需要做说明)。一旦标记在给出的通道中达到“标记号“,继续进行加工，不结束准停。
WAITE (n,n)	等待给出的通道的程序结尾(自身通道不作说明)。
SETM (标记号,标记号,…)	在自身的通道里设定标记 “标记号” 对正在进行的处理没有影响。SETM（）跳过RESET和NC-START 保存有效性。 SETM（）也可以从同步中被编程。
CLEARM (标记号,标记号,…)	在自身的通道里取消标记 “标记号” 对正在进行的处理没有影响。通道中的所有标记都可以用 CLEARM（）取消。CLEARM（0）取消标记“0“。CLEARM（）跳过 RESET 和 NC-START 保存有效性。 CLEARM（）也可以从同步中被编程。



说明

以上所有的命令都必须在独立的程序段中。
标记的数量取决于装入的 CPU。

通道名

通道名必须从变量(详见第十章“变量和计算参数“)变成数字。



数字赋值应当在轻率的修改前被保存。

比如:

以 “MASCHINE“ 为名的通道应当保存通道号 1。

以 “LADER“ 为名的通道应当保存通道号 2：

```
DEF INT MASCHINE=1, LADER=2
```

变量保存与通道相同的名称：

由此比如说指令显示 START:

```
START(MASCHINE)
```

程序协调实例

通道 1:

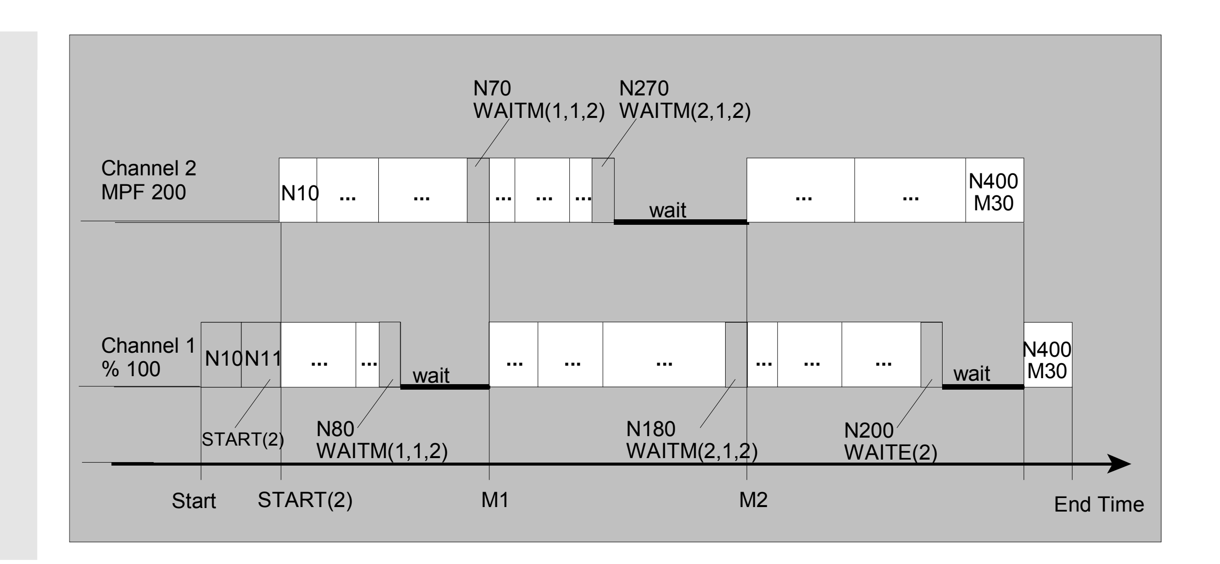
```
%_N_MPF100_MPF
```

N10 INIT(2,"MPF200")	
N11 START(2) .	在通道 2 中执行
N80 WAITM(1,1,2) .	等待 WAIT-通道 1 和通道 1 中其它执行的通道 2 中的标记 1
N180 WAITM(2,1,2) .	等待 WAIT-通道 1 和通道 1 中其它执行的通道 2 中的标记 2
N200 WAITE(2)	等待通道 2 的程序结尾
N201 M30 …	通道 1 程序结尾,总的结尾

通道 2：

```
%_N_MPF200_MPF
```

;$PATH=/_N_MPF_DIR	
 N70 WAITM(1,1,2) .	在通道 2 中执行 等待 WAIT-通道 1 和通道 1 中其它执行的通道 2 中的标记 1
N270 WAITM(2,1,2) .	等待 WAIT-通道 1 和通道 1 中其它执行的通道 2 中的标记 2
N400 M30	通道 2 的程序结尾



来自工件的程序实例

```
N10 INIT(2,"/_N_WKS_DIR/_N_WELLE1_WPD/_N_ABSPAN1_MPF")
```

带有相对路径的**Init**命令实例

```
;在通道 1 中选择程序 /_N_MPF_DIR/_N_MAIN_MPF 。
N10 INIT(2,"MYPROG") ; 在通道2中选择程序 /_N_MPF_DIR/_N_MYPROG_MPF 。
```



其它说明

对于程序间的数据交换,那些通道共同支配的变量可以
被使用 (NCK-专用的全局变量)。
其它情况中每个通道的程序都是被分开建立的。



版本 **SW 3** 以前

在开始命令之后 WAITE
不能立刻被询问,因为否则在这个程序开始之前,就识
别出程序结束。

纠正　编程一个停留时间。

比如:

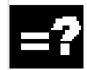
```
N30 START(2)
N31 G4 F0.01
N40 WAITE(2)
```

1.14 中断程序



编程

```
SETINT (3) PRIO=1 NAME
SETINT(3) PRIO=1 LIFTFAST
SETINT(3) PRIO=1 NAME  LIFTFAST
G… X… Y… ALF=…
DISABLE (3)
ENABLE (3)
CLRINT (3)
```



命令解释

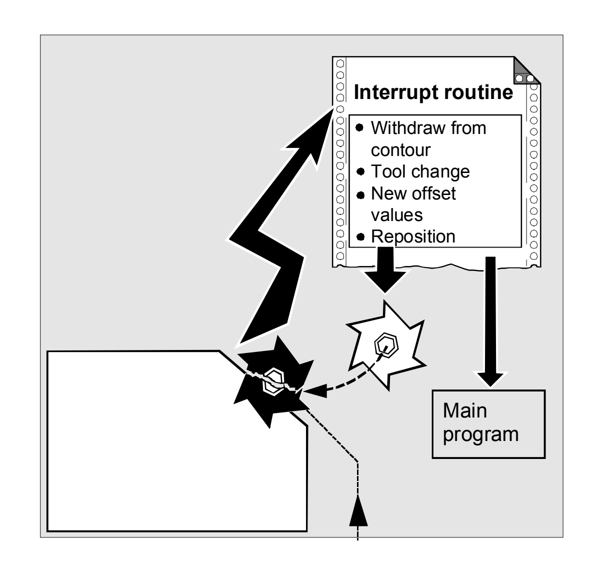
SETINT(n)	启动中断程序，当输入端n接通时, n (1…8) 是输入端序号。
PRIO=1	确定优先级别 1 到 128 (1 最高)。
LIFTFAST	快速离开工件轮廓
NAME	这里是被执行的子程序的的名称
ALF=…	可编程的运行方向 (在运动程序段中)
DISABLE(n)	切断中断程序，序号 n
ENABLE(n)	再次接通中断程序，序号 n
CLRINT(n)	取消中断程序序号 n 的中断赋值

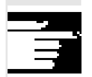


功能

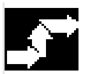
举例:在加工过程中工具折断。由此触发一个信号,这个信号中止正在运行的处理过程并同时开始一个子程序,也就是那个所谓的中断程序。在这个子程序中有所有在这种情况下应当被执行的指令。

如果这个子程序被执行(由此进入待命状态),控制跳回主程序并且根据 REPOS-命令继续从上次中断的地方处理。





REPOS 的更多信息参见第九章
轨迹运行特性,返回到轮廓。



运行

生成作为子程序的中断程序

这个中断程序在定义时和一个子程序一样被标识。

比如:

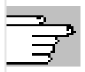
```
PROC ABHEB_Z
N10…
N50 M17
```

程序名 ABHEB_Z, 然后是 NC-程序段,最后以 M17 结束并回到主程序。



提示:

在中断程序内, SETINT 指令可以被编程并由此立即接通其它的中断程序。只有通过输入端才可以触发。



更多关于子程序的生成详见第二章。

中断位置保存, SAVE

中断程序可以在定义时用 SAVE 标识

比如:

```
PROC ABHEB_Z SAVE
N10…
N50 M17
```

通过 SAVE 定语,模态 G 功能会在中断程序结束后被调节到中断程序开始时的值。此外对于可调节的零点偏置(模态 G 功能组 8),
可编程的零点偏置和基准偏置再次被生成。如果G 功能组 15 (进给类型)产生变化,比如从 G94 到 G95, 相应的 F 值也会再次被生成。

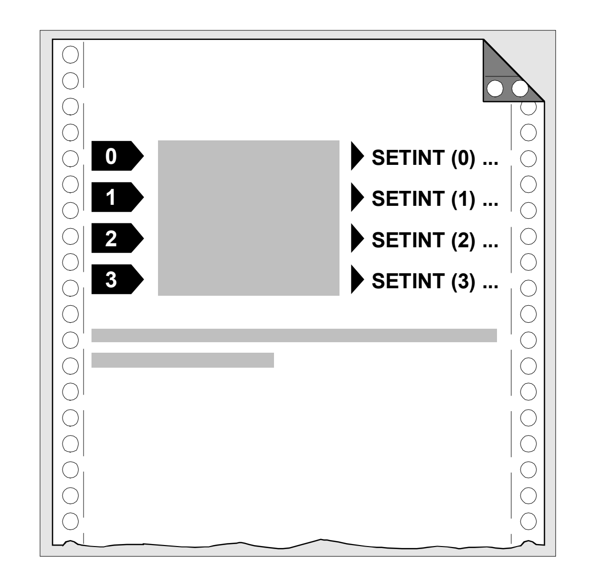
由此以后的处理可以从中断处继续进行。

中断程序赋值和开始, SETINT
控制系统支配信号(输入端1…8)，它能引起正在进行的程序的中断和启动相应的中断程序。

哪一个输入端分配到哪一个程序，在执行程序中进行。

比如:
```
N10 SETINT(3) PRIO=1 ABHEB_Z
```

在接通输入端 3 时程序 ABHEB_Z 立刻被启动。



启动更多中断程序,确定级别, PRIO=
如果在您的 NC-程序中有多个 SETINT 指令并且由此多个信号同时输入,您必须确定中断程序的级别,据此进行处理。优先级 1 到 128, 1最高。

比如:
```
N10 SETINT(3) PRIO=1 ABHEB_Z
N20 SETINT(2) PRIO=2 ABHEB_X
```

如果多个输入端同时保留,程序会根据级别数的顺序进行处理。首先SETINT(3),然后SETINT(2)。

如果在中断处理期间有新的信号输入,有较高优先级的程序中断当时的中断程序。

终止/再次启动中断程序, DISABLE, ENABLE
您可以在 NC 程序中以 DISABLE(n) 终止中断程序并可以用 ENABLE(n) 再次接通 (n 是输入端号)。

输入端/程序赋值在 DISABLE 时保持，并可以用 ENABLE 再次激活。

中断程序重新赋值

如果一个确定的输入端被一个新的程序赋值,旧的值自动失效。

比如:

```
N20 SETINT(3) PRIO=2 ABHEB_Z
...
...
N120 SETINT(3) PRIO=1 ABHEB_X
```

取消赋值, CLRINT

用 CLRINT(n) 可以取消赋值。

比如:

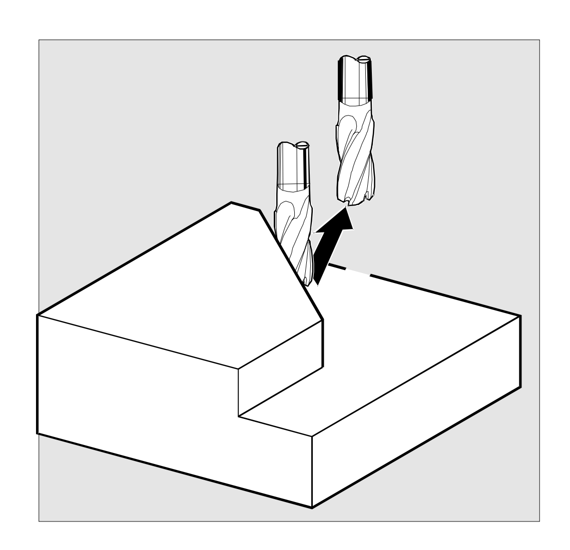
```
N20 SETINT(3) PRIO=2 ABHEB_Z
N50 CLRINT(3)
```

输入端 3 和程序 ABHEB_Z 之间的赋值被取消。

快速离开工件轮廓

用 LIFTFAST

在接通一个输入端时，工具会通过快速离开工件轮廓离开。



如果 SETINT 指令除了 LIFTFAST 还包含一个中断程序,快速离开会在中断程序之前被执行。

比如:

```
N10 SETINT(2) PRIO=1 LIFTFAST

N30 SETINT(2) PRIO=1 ABHEB_Z LIFTFAST
```

在两种情况中,在接通有最高优先权的输入端2时快速离开被执行。

- 在 N10 的情况下执行过程会被警报 16010 中断(因为没有说明异步的子程序 ASUP)。
- 在 N30 的情况下执行 ASUP "ABHEB-Z"。

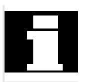
在确定离开方向时会检测,是否有一个框架带镜像被激活。在这种情况下,刀具沿正切线离开,左右相间。在刀具方向的方向分量没有镜像。这种操作通过 MD $MC_LIFTFAST_WITH_MIRROR=TRUE 被激活

快速离开的运动过程
几何轴快速离开工件轮廓时所移动的距离，可以在机床数据中设定。

没有 LIFTFAST 的中断程序
一旦在轨道上的运行停止,就要在轨道上制动并启动中断程序。
这个位置会作为中断位置保存起来并在 REPOS 时以 RMI 在中断程序结束后启动。

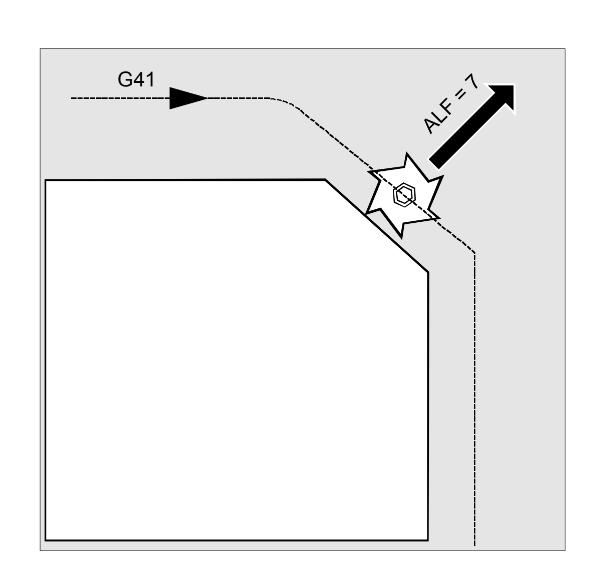
带 LIFTFAST 的中断程序
在轨道上刹车并同时进行 FIFTFAST
运动作为迭加运动。如果轨道运动和 LIFTFAST
运动停止,中断程序被启动。
作为中断位置在轮廓上的位置被保存,在这个位置上开始 LIFTFAST 运动并由此离开轨道。

版本 SW 5.1 以上带 LIFTFAST 和 ALF=0
的中断程序与不带 LIFTFAST 的中断程序相同。



可编程的运行方向, ALF=…

在 NC 程序中，给出工具快速离开时的方向。



可能的运行方向存储在控制系统中，带专门的代码号，并可以在这个代码下调用。

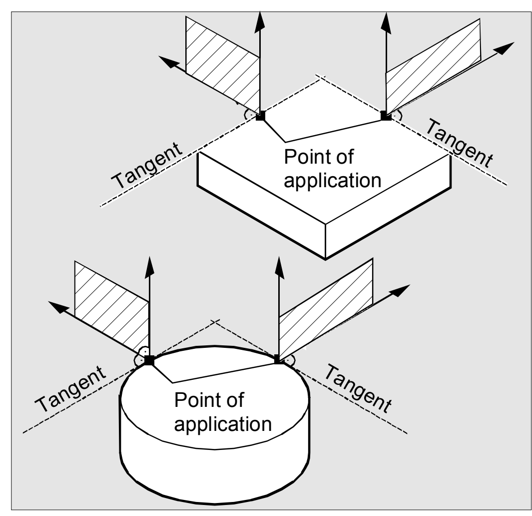
比如:

```
N10 SETINT(2) PRIO=1 ABHEB_Z  LIFTFAST
ALF=7
```

带接通的 G41
(加工方向工件轮廓左边)的工具从俯视的角度看垂直离开工件轮廓。

运行方向的基准面

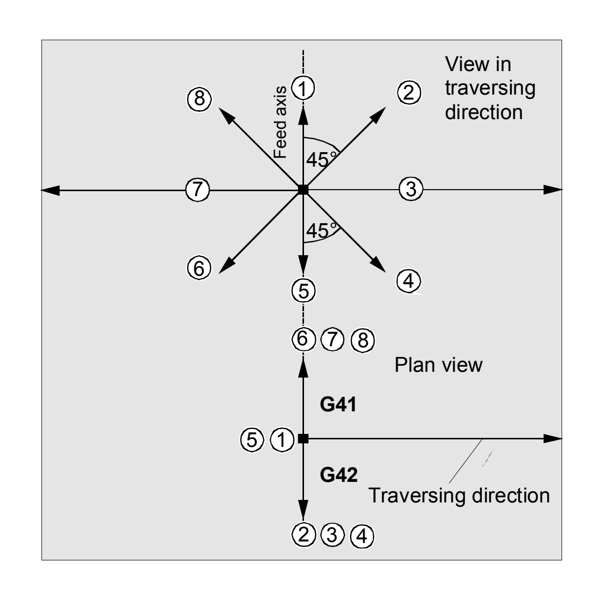
工具在编程的轮廓上的切入点有一个平面,它作为带相应代码离开运动的参数说明的基准面。



这个基准面由工具径向轴(进刀方向)和一个矢量组成,这个矢量与这个平面相对并与工具在轮廓上的切入点的切线垂直。

代码序号概述，带运行方向

从这个基准面出发您可以在旁边的图象里找到带运行方向的代码序号。



用 ALF=0 停止快速离开功能。



注意:

在进行工具半径补偿时不应当
在 G41 时使用编码 2, 3, 4 以及
在 G42 时使用编码 6, 7, 8。

在这些情况下工具驶向轮廓并会与工件相撞。

自软件版本 SW 4.3 起的撤回运动

撤回运动的方向通过 G 代码 **LFTXT** 或带变量 **ALF** 的 **LFWP** 被编程。

- **LFTXT**
  撤回运动的平面由轨道切线和工具方向确定。通过这种 G 代码(标准设定),
  编程快速离开时之前的性能。
- **LFWP**
  这个撤回运动的平面是有效的工作平面,这个平面通过 G 代码 G17, G18, 或者 G19
  选择。撤回运动的方向不由轨道切线决定。由此可以编程一个与轴并行的快速离开。
- **LFPOS (版本 SW 5 以上)**
  用 POLFMASK 标明的轴撤回到用 POLF
  编程的绝对轴位置。详见功能描写 M3 中 NC
  进行的撤回。
  对于多个轴（版本 SW 6 以上)
  以及多个线性关系的轴(版本 SW 7 以上)，ALF
  对离开方向没有影响。

在撤回运动平面中与前面一样，用 ALF 编程方向，以 45 度不连续的步骤。 在 **LFTXT** 时对于 ALF=1
撤回方向为工具方向。

在 **LFWP** 时工作平面中的方向由以下赋值决定:

- **G17**: X/Y平面 ALF=1 在 X 方向撤回
  ALF=3 在 Y 方向撤回
- **G18**: Z/X平面 ALF=1 在 Z 方向撤回
  ALF=3 在 X 方向撤回



- **G19**: Y/Z平面 ALF=1 在 Y 方向撤回
  ALF=3 在 Z 方向撤回



编程举例

在这个例子中，折断的刀具自动地被另一个刀具替代。
加工以新的刀具继续进行。

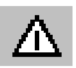
主程序

N10 SETINT(1) PRIO=1 W_WECHS -> -> LIFTFAST	如果输入端 1 接通,刀具会立刻以快速离开(代码7对应工具半径补偿 G41) 的形式离开工件轮廓。然后中断程序 W_WECHS 被执行。
N20 G0 Z100 G17 T1 ALF=7 D1	
N30 G0 X-5 Y-22 Z2 M3 S300	
N40 Z-7	
N50 G41 G1 X16 Y16 F200	
N60 Y35	
N70 X53 Y65	
N90 X71.5 Y16	
N100 X16	
N110 G40 G0 Z100 M30	

子程序

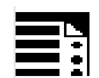
PROC W_WECHS SAVE	带当前运行状态储存的子程序
N10 G0 Z100 M5	工具更换位置,主轴停止
N20 T11 M6 D1 G41	更换工具
N30 REPOSL RMB M3	再次启动轮廓并跳回主程序

->在一个程序段中编程。



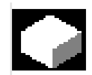
*如果您在子程序中没有编辑 REPOS-命令,
这样就会定位在程序段的结尾,这个程序段跟在那个被
中断的程序段后面。*

1.15 交换轴,交换主轴



命令解释

`RELEASE(Achsname, Achsname, …)`	轴使能
`GET(Achsname, Achsname, …)`	轴接收
`GETD  (Achsname, Achsname, …)`	轴直接接收
`Achsname`	在系统中轴赋值: AX1, AX2, … 或者给出加工轴名称
`RELEASE(S1)`	主轴 S1,S2,…的使能
`GET(S2)`	主轴 S1,S2,…的接收
`GETD(S3)`	主轴 S1,S2,…的直接接收

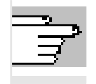


功能

一个或多个轴和主轴可以在一个通道中被使用。如果一个轴必须交替地在两个不同的通道里工作（比如托盘交替），必须首先在当前的通道使能，然后接收到另一个通道。轴会在两个通道之间进行转换。



更多变换轴或主轴的功能的信息详见
/FB/, K5 BAGs, 通道,轴变换



运行

轴交换的前提

- 轴必须通过机床数据在所有使用这些轴的通道中定义。
- 关于轴专用的机床数据必须确定在 POWER ON（上电）之后，轴应该在哪一个通道中赋值。

轴使能: RELEASE

在轴使能时必须要注意:

1. 轴不可以参加转换。
2. 在轴耦合时(正切控制),所有相关轴都必须使能。
3. 一个角逐的定位轴在这种状态下不能交换。
4. 在龙门架主轴机床中，所有跟随轴也被交换。
5. 在轴耦合时(联动,引导轴耦合,电子齿轮)只有相连的引导轴被使能。

轴接收: GET

用这个命令执行原来的轴交换。轴的职责完全取决于通道,在这个通道中编程了该命令。

GET 的作用

带同步的轴变换:

假如一个轴间断的在另一个通道中赋值，或者被PLC赋值,并且在 GET 之前不通过 “WAITP“、G74 或取消剩余行程使之同步，则该轴必须始终同步。

- 停止进刀(与 STOPRE 相同)
- 加工停止，直至交换完成为止。

不同步的轴变换:

如果轴不必同步,则 GET 不产生进刀停止。

比如:

```
N01 G0 X0
N02 RELEASE(AX5)
N03 G64 X10
N04 X20
N05 GET(AX5)        如果不需要同步,则不产生可执行的程序段。
N06 G01 F5000       没有可执行程序段。
N07 X20             不是可执行程序段,因为X轴位置与在 N04 时一样。
N08 X30             N05 之后第一个可执行程序段。
N09 …
```

自动的 "GET"

如果一个轴在通道中原则上可用,但是当时实际上不是作为"通道轴",就会有一个 GET 自动被执行。

如果这个(些)轴已经被同步,就不会产生进刀停止。



一个用GET接收的轴也会在按键之后或程序复位之后保持该通道的赋值。如果这些轴在它们的主通道里是必要的,则在程序新启动时,变换的轴或主轴必须进行程序的赋值。

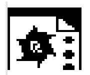
在 POWER ON（上电）后,

它将给在机床数据中保存的通道赋值。

轴直接接收: GETD

用GETD (GET Directly)

将一个轴从另一个通道中直接取出。这意味着,不必有合适的 RELEASE 在另一个通道中为这个 GETD 编程。这也意味着,现在必须建立另一个通道通讯(比如等待符)。



编程举例

6 个轴在通道1中用于加工的为：1., 2., 3. 和第 4. 轴。
5. 和第 6. 轴用于通道 2 中进行工件更换。
轴 2 应当在两个轴之间可以进行交换并在 POWER ON 之后给通道1赋值。

通道1中的程序 “MAIN“

%_N_MAIN_MPF

INIT (2,"TAUSCH2")	在通道 2 中选择程序 TAUSCH2
N… START (2)	启动通道 2 中的程序
N… GET (AX2) … …	接收轴 AX2
N… RELEASE (AX2)	使能轴 AX2
N… WAITM (1,1,2)	在通道 1 和 2 中等待等待符以便在两个通道中实现同步。
N… N… M30	轴变换之后的流程

通道 2 中的程序“变换 2“

%_N_TAUSCH2_MPF

N… RELEASE (AX2)	
N160 WAITM (1,1,2)	在通道 1 和 2 中等待等待符以便在两个通道中实现同步。
N150 GET (AX2)	接收轴 AX2
N… N…M30	轴变换之后的流程

轴变换性能更改设定

轴交换的时间点由 MD10722：AXCHANGE_MASK 按以下方法设定：

- 如果轴通过 WAITP 处于一个中性状态(与前面的性能一样),那么也可以在两个通道之间进行自动的轴变换。
- 版本 **SW 5.3** 以上,所有用 GET 或者 GETD 取到轴容器中的轴，在轴容器旋转以后才可以再次被变换。
- 版本 **SW 6.4** 以上,在插入一个中间程序段之后会在主程序中检测,是否需要重组。只有当这个程序段的轴状态和实际的轴状态不相符时,才需要进行重组。



编程举例

激活轴交换，没有进刀停止

N010 M4 S100	
N011 G4 F2	
N020 M5	
N021 SPOS=0	
N022 POS[B]=1	
N023 WAITP[B]	轴 B 变成中性轴
N030 X1 F10	
N031 X100 F500	
N032 X200	
N040 M3 S500	
N041 G4 F2	
N050 M5	
N099 M30	

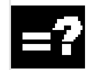
如果主轴(轴 B)在程序段 N023 之后作为 **PLC** 轴处理,比如运行到 180 度角,再返回到 1 度,然后再成为中性轴，则程序段 N040 没有引起进刀停止，并且无需重组。

1.16 NEWCONF:有效设置机床数据(版本 SW 4.3 以上)



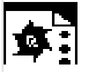
功能

用语言指令 NEWCONF, “NEW_CONFIG“
有效级的所有机床数据都被设置为有效。功能与按软
键 “MD 有效设置“相对应。
在运行 NEWCONF 功能时会出现一个隐含的进刀停止,
也就是说轨道运行会停止。



解释

NEWCONF	有效级别 ”NEW_CONFIG” 的所有机床数据设置为有效。

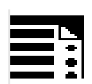


编程举例

铣削加工:用不同的工艺加工钻孔的位置。

```
N10 $MA_CONTOUR_TOL[AX]=1.0                ;更改机床数据
N20 NEWCONF                                ;有效设置机床数据
```

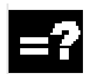
1.17 WRITE:写文件（自软件版本 SW 4.3 起)



编程

```
WRITE(var int error, char[160] filename, char[200] string)
```

WRITE 命令在给出的文件结尾附加一个程序段。



参数解释

error	错误变量的返回	
	0	没有错误
	1	路径不允许
	2	路径没找到

	3	文件没找到
	4	错误的文件类型
	10	文件已满
	11	文件被使用
	12	没有空余资源
	13	没有存取权限
	20	其它错误

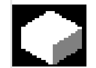
`filename`

字符串写入的文件名称包含文件名空格或者控制符号(十进制符号)
ASCII-代码 <= 32), 这样WRITE-命令会被错误提示 1
“路径不可用“中断。
文件名可以用路径和文件标识给出。文件名必须是绝对的,也就是说它以“
/“开头。如果文件名不包含范围标记 (_N_),
它会被相应的填满。如果没有给出标记 (_MPF或_SPF),
就会被自动填上_MPF。若给出的信息没有路径,文件会存放在当前的目
录(＝选中程序的目录)中。文件名的长度最多可以有 32
个字节,路径的长度最多可以有 128 个字节。

举例: _PROTFILE
_N_PROTFILE
_N_PROTFILE_MPF
/_N_MPF_DIR_/_N_PROTFILE_MPF/

`string`

正在写入的文本。会在内部附加LF,也就是说这个文本会变长一个字符。



功能

用 WRITE
命令可以在给出的文件结束处附加文件(比如测量循
环时的测量结果)。
通过 MD 11420 LEN_PROTOCOL_FILE
可以把协议文件的最大长度调节到千字节。这个长度
对于所有用 WRITE 命令设定的文件都有效。

如果文件达到给定的长度,就会出现一个出错提示,字
符串不会被保存。如果存储器够用,则可以编制一个新
的文件。

所编制的文件可以

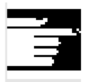
- 由所有的使用者读、修改和删除，
- 写入到正在执行的零件程序中。

程序段在文件结束处插入，也就是说在 M30 之后。



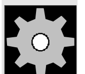
编程举例

N10 DEF INT ERROR	;
N20 WRITE(ERROR,"TEST1","PROTOKOLL VOM 7.2.97")	;从记录 7.2.97 把文本写入文件 TEST1 中
N30 IF ERROR	;
N40 MSG ("Fehler bei WRITE-Befehl:" <<ERROR)	;
N50 M0	;
N60 ENDIF	;
...	
WRITE(ERROR, "/_N_WKS_DIR/_N_PROT_WPD/_N_PROT_MPF", "PROTOKOLL VOM 7.2.97")	;给出绝对路径



其它说明

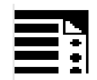
- 一个用 WRITE 命令描述的文件被重新编制,如果它不存在于 NC 中的话。
- 如果硬盘中有一个相同名称的文件,则文件关闭后(在 NC 中)被覆盖。
  纠正在“通讯”操作范围中通过软键“性能”修改 NC 中的名称。



机床生产商

用 WRITE 命令可以从零件程序中存放到文件中。
记录文件(千字节)的大小在 MD 中确定。

1.18 DELETE:删除文件(版本 SW 4.3 以上)



编程

```
DELETE(var int error, char[160] filename)
```

DELETE-命令 删除给出的文件。



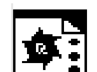
参数解释

error	错误变量的返回 0 没有错误 1 路径不可用 2 路径没找到 3 文件没找到 4 错误的文件类型 11 文件被使用 12 没有空余资源 20 其它错误
filename	应当被删除的文件名称。 文件名可以用路径和文件标识给出。路径名必须是绝对的,也就是说它以"/"开头。如果文件名不包含范围 (_N_), 它会被相应的填满。 文件标识("_"加 3 个字符),比如: _SPF)是任选的。 没有标识时,文件名会被自动加上 _MPF。若给出的信息没有路径,文件会存放在当前的目录(选中程序的目录)中。文件名的长度最多可以有 32 个字节,路径的长度最多可以有 128 个字节。
	比如: PROTFILE _N_PROTFILE _N_PROTFILE_MPF /_N_MPF_DIR/_N_PROTFILE_MPF/



功能

用 DELETE 命令可以删除所有的文件, 无论它是否通过
WRITE 命令产生。
通过更高存取级别产生的文件可以用 DELETE 删除。

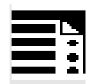


编程举例

N10 DEF INT ERROR	;
N15 STOPRE	; 进刀停止
N20 DELETE (ERROR,	; 在子程序中删除文件 TEST1

```
            "/_N_SPF_DIR/_N_TEST1_SPF")
N30 IF ERROR                              ;
N40   MSG ("Fehler bei DELETE-Befehl:"    ;
                               <<ERROR)
N50   M0                                  ;
N60 ENDIF                                 ;
...
```

1.19 READ:读取文件中的行(版本 SW 5.2 以上)



编程

```
READ(var int error, string[160] file, int line, int number, var
string[255] result[])
```

READ-命令在给出的文件中读取一行或多行,并把读取到的内容存储在STRING类型的数组中。
每被读取的行都占用一个数组元。

参数解释

`error`	错误变量的返回 (Call-By-Reference 参数,类型 INT) 0 没有错误 1 路径不可用 2 路径没找到 3 文件没找到 4 错误的文件类型 13 存取权限不够 21 不是当前行(参数 ”line” 或 ”number” 比文件的行数大) 22 结果变量“结果“的数组长度过小 23 行范围过大(参数“数字“选择过大,以至超出文件结束)。
`file`	用于读取的文件的名称/路径(最大长度为 160 字节的 STRING 类型 Call-By-Value-参数)文件必须位于 NCK (未有效的文件系统)的用户存储器中。文件名称之前可以放置范围标识_N_。缺少的范围标识会被相应地填充。 文件标识 (“_”加 3 个字符,比如: _SPF)是任意的。如果实际上没有标识,文件名会自动被加上 _MPF。 如果在“文件“中没有路径说明,将在当前目录(=所选程序的目录)寻找文件。 现有的“文件“中的路径说明必须以“/“开头(绝对路径)。

line	给出要读取的行范围的位置(INT 类型的 Call-By-Value 参数) 0　用参数“数字“给出 　读文件结束前的行数。 1 到 n　第一个要读取的行的数字。
number	要读取的行的个数(INT 类型的 Call-By-Value 参数)。

Result	字符串类型的数组，在这个数组中存放被读取的文本。 (长度为 255 字节的 Call-By-Reference-参数)。



功能

用命令 READ 您可以从一个文件中读取一行或
数行信息。
被读取的行会被存储在一个数组的数组元中。这些信息
以 STRING 出现。



其它说明

- 二进制文件不能被读入。给出错误 error=4:
  错误的文件类型。以下的文件类型不可读: _BIN,
  _EXE, _OBJ, _LIB, _BOT, _TRC, _ACC, _CYC,
  _NCK。
- 当前设定的保护级别必须与文件的 READ
  权限相等或大于它。如果不时这种情况,存取会被
  error=13 拒绝。
- 如果在参数“数字“中给出的行数比“结果“的数组长
  度小,剩下的数组元不会被修改。
- 通过控制符号 "LF" (Line Feed) 或 "CR LF"
  (Carrige Return Line Feed),
  一行的结束不会被存放到“结果”目标变量中。如果
  被读取的行长于“结果“目标变量的字符串,它就会
  被切断。不会出现错误提示。

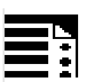


编程举例

```
N10 DEF INT ERROR                    ; 错误变量
N20 STRING[255] RESULT[5]            ; 结果变量
```

`...`	
`N30 READ(ERROR, "TESTFILE", 1, 5, RESULT)`	;没有范围标识和文件标识的文件名
`...`	
`N30 READ (ERROR, "TESTFILE_MPF", 1, 5, RESULT)`	;没有范围标识和文件标识的文件名
`...`	
`N30 READ(ERROR,"_N_TESTFILE_MPF",1,5, RESULT)`	;带范围标识和文件标识的文件名
`...`	
`N30 READ(ERROR,"/_N_CST_DIR/N_TESTFILE _MPF", 1, 5 RESULT)`	;带范围标识,文件标识和路径说明的文件名
`^...`	
`N40 IF ERROR <>0`	;错误计值
`N50 MSG("FEHLER "<<ERROR<<" BEI READ-BEFEHL")`	
`N60 M0`	
`N70 ENDIF`	
`...`	

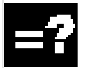
1.20 ISFILE:现存的用户存储器 NCK 中的文件(版本 SW 5.2 以上)



编程

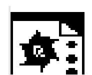
```
result=isfile(string[160]file)
```

用 ISFILE-命令 可以检测在 NCK (无源文件系统)的用户存储器中是否存在文件。作为结果会出现 TRUE (存在文件)或者 FALSE (不存在文件)。



参数解释

`file`	用于读取的文件的名称/路径(最大长度为 160 字节的 STRING 类型 Call-By-Value-参数) 文件必须位于 NCK (无源的文件系统)的用户存储器中。范围标识 _N_ 可以反放在文件名前面。缺少的范围标识会被相应地填充。 文件标识(“_”加3个字符,比如: _SPF) 是任意的。如果实际上没有标识,文件名会自动被加上 _MPF。 如果在“文件“中没有路径说明,将在当前目录(=所选程序的目录)寻找文件。现有的“文件“中的路径说明必须以“/“开头(绝对路径说明)。
`result`	用于接收 BOOL 类型结果的变量 (TRUE 或者 FALSE)



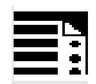
编程举例

```
N10 DEF BOOL RESULT
N20 RESULT=ISFILE("TESTFILE")
N30 IF(RESULT==FALSE)
N40   MSG("DATEI NICHT VORHANDEN")
N50   M0
N60 ENDIF
...
```

或者：

```
N30 IF(NOT ISFILE("TESTFILE"))
N40   MSG("DATEI NICHT VORHANDEN")
N50   M0
N60 ENDIF
...
```

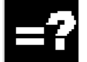
1.21 CHECKSUM:通过一个数组构成检查和（自软件版本 SW 5.2）



编程

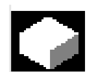
```
error=CHECKSUM(var string[16] chksum,string[32]array, int first, int last)
```

功能 CHECKSUM（检查和）通过数组构成检查和。



参数解释

error	错误变量的返回	描述
	0	没有错误
	1	没有找到符号
	2	没有数组
	3	变址 1 过大
	4	变址 2 过大
	5	无效数据类型
	10	检查和溢出
chksum	通过数组表示的检查和，作为字符串(字符串类型的 Call-By-Reference-参数,确定长度 16)。 检查和作为 16 进制的符号串被显示。但是不说明格式符号。 比如: "A6FC3404E534047C"	
array	数组名称，通过数组构成检查和。(长度最多为 32 的字符串类型 Call-By-Value-参数)。 容许的错误: 单个尺寸或者2维尺寸数组类型 BOOL, CHAR, INT, REAL, STRING 机床数据的数组不允许。	
first	开始栏目的栏目号(可选)	
last	结束栏目的栏目号(可选)	



功能

使用 CHECKSUM，通过数组构成检查和。

切削使用:

检测导入轮廓是否改变。



其它说明

参数 `first` (第一）和 `last`（最后）可选。

如果没有说明栏目变址,检查和通过整个数组构成。

检查和的结果总是明确的。修改数组元时也会产生另一个结果字符串。



编程举例

```
N10 DEF INT ERROR
N20 DEF STRING[16] MY_CHECKSUM
N30 DEF INT MY_VAR[4,4]
N40 MY_VAR=...
N50 ERROR=CHECKSUM
    (CHECKSUM;"MY_VAR", 0, 2)
...
```

在MY_CHECKSUM中提供值
"A6FC3404E534047C"

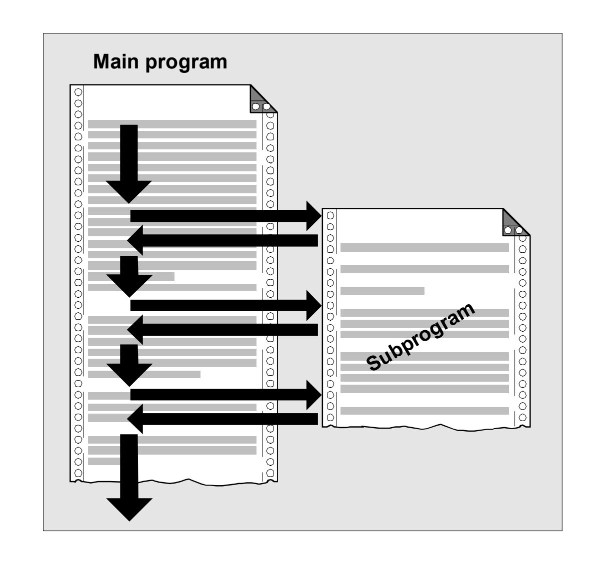
■

子程序技术，宏指令技术

2.1 使用子程序

子程序是什么？

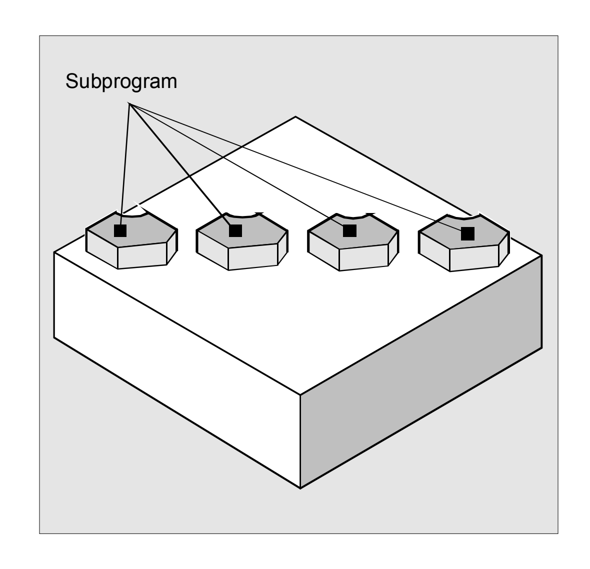
原则上讲，一个子程序的结构与一个零件程序一样。它由带运行指令和开关指令的 NC 程序段组成。



从本质上说，主程序与子程序没有区别。子程序中包含了要多次运行的工作过程或者工作步骤。

使用子程序

总是反复出现的加工步骤在子程序中仅编程一次。比如说某个确定的轮廓，它们总是反复出现，或者是一个加工循环。



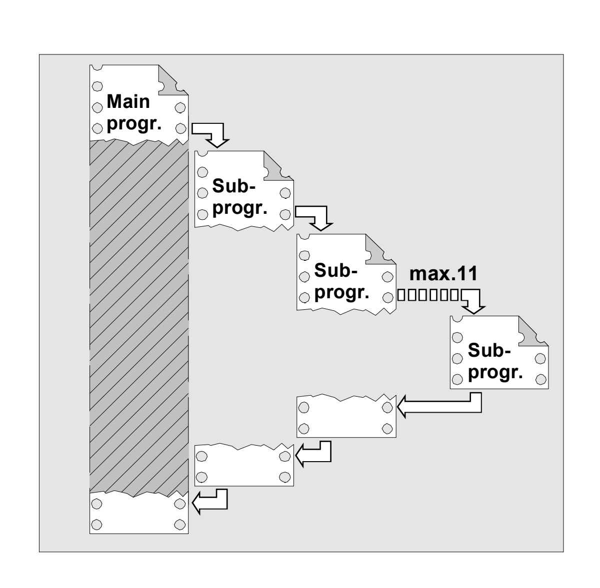
子程序可以在任意一个主程序中调用和执行。

子程序结构

子程序结构与主程序结构相同。

此外，在子程序中可以编程一个程序头，带参数定义。

嵌套深度



子程序嵌套深度

在一个子程序中可以再次调用一个子程序。在该子程序中再次有一个子程序调用，等等。
程序级面或者嵌套深度的最大个数可以为 12。

这表明：
从一个主程序中可以调用11个嵌套的子程序。

限制

即使在中断程序中，也可以调用子程序。对于中断程序中的工作，您应该保留 4 个级面，或者仅仅嵌套 7 个子程序调用。

对于西门子的加工循环和测量循环需要 3 个级面。如果从一个子程序调用一个循环，则可以最多在级面 5 中进行（如果 4 个级面为中断程序保留）。

2.2 子程序，带 SAVE 功能



功能

在带 PROC 的定义指令中，您要另外给出指令 SAVE 。

通过 SAVE 定语,模态 G 功能会在中断程序结束后被调节到中断程序开始时的值。如果由此产生一个 G 功能组 8（可设定零点偏移）的改变，G 功能组 52（一个可旋转工件的框架旋转）的改变，或者 G 功能组 53 （在刀具方向旋转框架） 的改变，则恢复当时的框架。

- 在子程序返回时，该有效的基准框架没有改变。
- 可编程的零点偏移被恢复。



自软件版本 **SW 6.1**
起，可设定零点偏移和基准框架的特性可以通过机床数据 MD 10617: FRAME_SAVE_MASK 改变。

相关的其它信息参见
/FB/ K1, 通用机床数据

举例：

子程序定义

```
PROC KONTUR (REAL WERT1) SAVE
N10 G91 …
N100 M17
```

主程序

```
%123
N10 G0  X… Y… G90
N20…
N50 KONTUR (12.4)
N60 X… Y…
```

在子程序 KONTUR 中，G91
增量尺寸生效。在返回到主程序中后，绝对尺寸再次生效，因为主程序的模态功能已经用 SAVE 存储。

2.3 子程序，带参数转让

程序开始, PROC

一个子程序，它在调用程序的程序运行时应该接收参数，该子程序用关键字 PROC 标记。

子程序结束 M17, RET

用指令 M17 标记子程序结束，
并且同时是返回到所调用主程序的指令。
可替换 M17 的：关键字 RET
位于子程序结束处，没有中断轨迹控制运行，并且没有到 PLC 的功能输出。

没有中断的轨迹控制运行

前提条件就是轨迹控制运行没有中断。
该子程序不允许有 SAVE 属性。有关 SAVE
功能的详细说明参见章节 2.2。



RET必须位于独立的程序段中编程。

举例：

```
PROC KONTUR
N10…
…
N100 M17
```

在主程序和子程序之间的参数传递

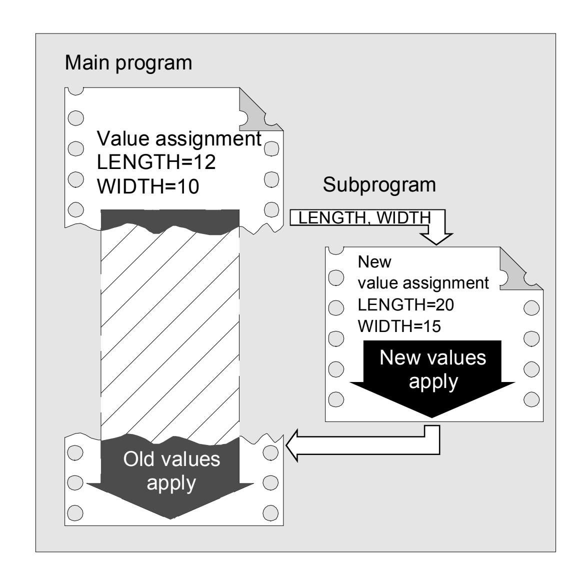
如果在主程序中带参数工作，则您也可以在子程序中使用相应计算的或者赋值的数值。

在此主程序的实际参数的值在子程序调用时传递到子程序的形式参数，并且在子程序执行过程中处理。

举例：

```
N10 DEF REAL LAENGE,BREITE
N20 LAENGE=12 BREITE=10
N30 RAHMEN(LAENGE,BREITE)
```

在主程序 N20 中赋值的数值传递到子程序 N30 中。
参数传送按照所给定的顺序进行。
参数名称在主程序和子程序中不可一样。



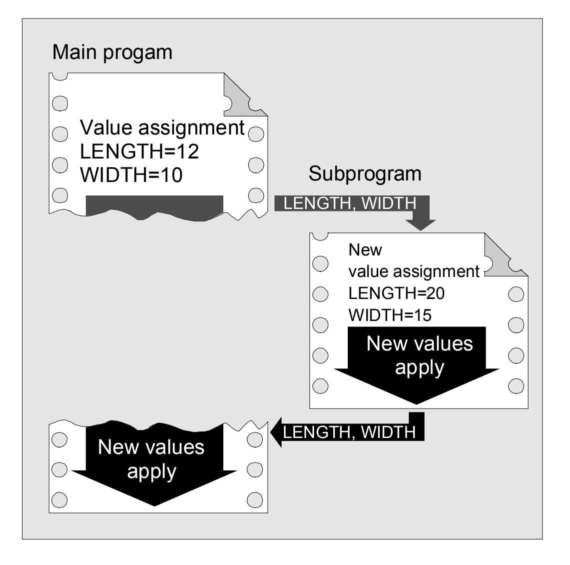
参数传送的**2**种方法

仅传送的值 (Call-by-value)

如果所传送的参数在执行子程序过程中改变，则这对主程序没有影响。这里参数保持不变（见图）。

参数传送，带数据交换 (Call-by-reference)

在子程序中参数每次改变均同时影响主程序中该参数的改变（见图）。





编程

参数传送中重要的参数必须在子程序开始时说明名称和类型。

参数传送 Call-by-value

```
PROC PROGRAMMNAME(VARIABLENTYP1 VARIABLE1,VARIABLENTYP2 VARIABLE2,…)
```

举例：

```
PROC KONTUR(REAL LAENGE, REAL BREITE)
```

参数传送 Call-by-reference,
用关键字 VAR 标记

```
PROC PROGRAMMNAME(VAR VARIABLENTYP1 VARIABLE1,VAR VARIABLENTYP2 …,)
```

举例：

```
PROC KONTUR(VAR REAL LAENGE, VAR REAL BREITE)
```

数组传送 Call-by-reference,
用关键字 VAR 标记

```
PROC PROGRAMMNAME(VAR VARIABLENTYP1 FELDNAME1[数组规格],
VAR VARIABLENTYP2 FELDNAME2[数组规格], VAR VARIABLENTYP3
FELDNAME3[数组规格1, 数组规格2], VAR VARIABLENTYP4 FELDNAME4[ ],VAR
VARIABLENTYP5 FELDNAME5 [,数组规格])
```

举例：

```
PROC PALETTE(VAR INT FELD[,10])
```



其它说明

带 PROC 的定义指令必须在一个独立的 NC 程序段中编程。可以最多有 127 个参数用于参数传送。



数组定义

对于形式参数的定义，适用于：

在二维数组中，第一个尺寸的数组个数不要说明，但是必须写逗号。

举例：

```
VAR REAL FELD[,5]
```

子程序可以用不确定的数组长度加工处理可变长度的数组。在定义变量时仍然必须要确定应该接收多少单元。

数组定义的说明可以在编程说明“工作准备部分”中找到。

编程举例

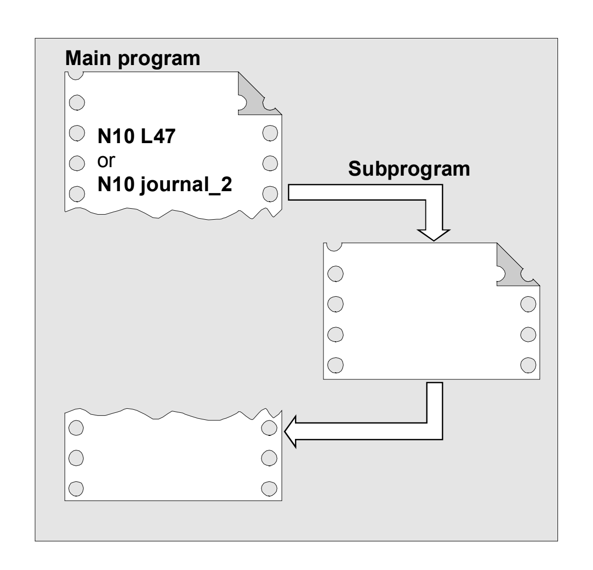
带可变数组长度的编程	
%_N_BOHRPLATTE_MPF	主程序
DEF REAL TABELLE[100,2]	定义位置表
EXTERN BOHRBILD (VAR REAL[,2],INT)	
TABELLE[0,0]=-17.5	确定位置
…	
TABELLE[99,1]=45	
BOHRBILD(TABELLE,100)	子程序调用
M30	

利用一个所传送的、可变长度的位置表建立一个钻孔图	
%_N_BOHRBILD_SPF	子程序
PROC BOHRBILD(VAR REAL FELD[,2],-> -> INT ANZAHL)	参数传送
DEF INT ZAEHLER	
STEP: G1 X=FELD[ZAEHLER,0]-> -> Y=FELD[ZAEHLER,1] F100	加工顺序
Z=IC(-5)	
Z=IC(5)	
ZAEHLER=ZAEHLER+1	
IF ZAEHLER<ANZAHL GOTOB STEP	
RET	子程序结束

2.4 子程序调用：L 或者 EXTERN

子程序调用 没有参数传送

在主程序中调用子程序时，可以使用地址L和子程序号，或者直接通过程序名称。



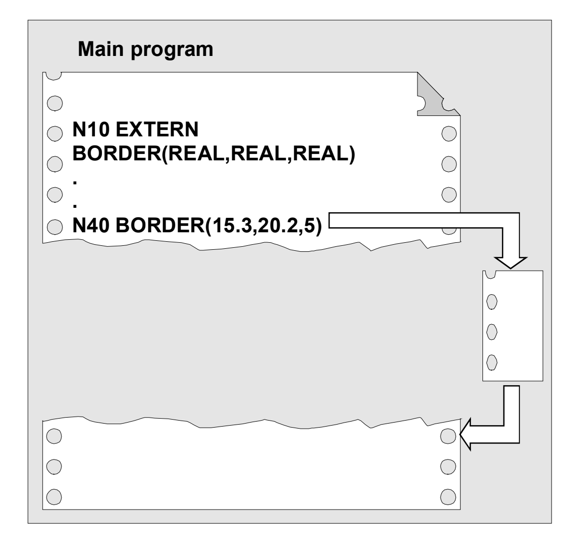
举例：

```
N10 L47 或者
N10 ZAPFEN_2
```

子程序，带参数传送,

EXTERN 说明

必须在调用有 EXTERN
的主程序之前引用带参数传送的子程序，
比如在程序开始处。
说明子程序名称，并且按照传送顺序说明变量类型。



只有当子程序在工件中或者在全局子程序目录下时才必须说明 EXTERN。
循环不可以作为 EXTERN 说明。

```
EXTERN-指令
EXTERN NAME(TYP1, TYP2, TYP3, …) 或者
EXTERN NAME(VAR TYP1, VAR TYP2, …)
```

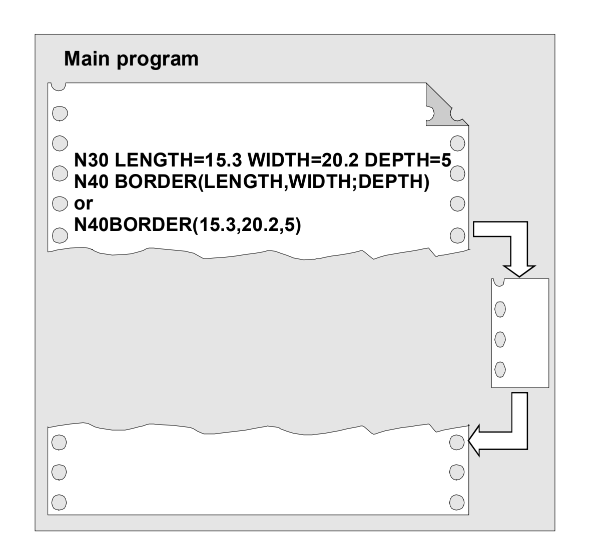
举例：

```
N10 EXTERN RAHMEN(REAL, REAL, REAL)
…
N40 RAHMEN(15.3,20.2,5)
```

N10 说明子程序，N40 调用带参数传送的子程序。

子程序调用，带参数传送

在主程序中，通过说明程序名和参数传送调用子程序。在参数传送时，您可以直接（不通过 VAR 参数）传送变量或者值。



举例：

```
N10 DEF REAL LAENGE,BREITE,TIEFE
N20 …
N30 LAENGE=15.3 BREITE=20.2 TIEFE=5
N40 RAHMEN(LAENGE,BREITE,TIEFE)
```

或者

```
N40 RAHMEN(15.3,20.2,5)
```

子程序定义符合子程序调用



不仅是变量类型，而且传送的顺序都必须与定义一致，该定义已经在子程序名中在 PROC 下约定。参数名称可以在主程序和子程序中不一样。

举例：

在子程序中定义：

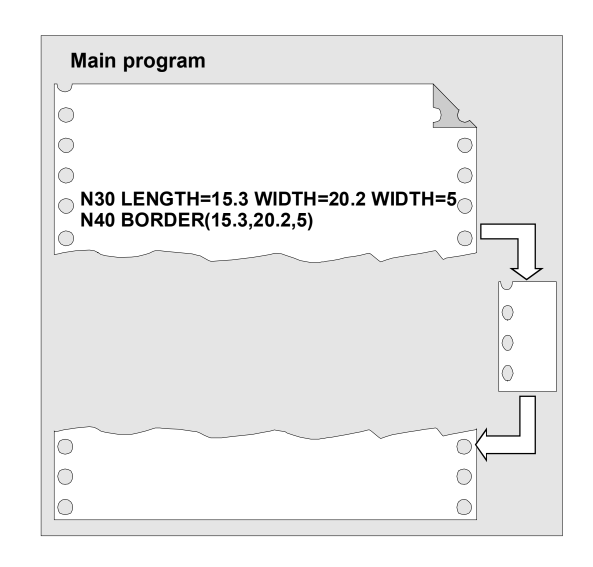
```
PROC RAHMEN(REAL LAENGE, REAL BREITE, REAL TIEFE)
```

在主程序中调用：

```
N30 RAHMEN(LAENGE, BREITE, TIEFE)
```

不完整的参数传送

在子程序调用时，可以删除自身规定的值或者参数。在这种情况下，用零占用子程序中相应的参数。



在说明顺序时必须写逗号。如果参数位于顺序的结束处，则可以同样取消该逗号。

返回到最后的示例：

```
N40 RAHMEN(15.3, ,5)
```

这里删除均值 20.2。

说明



类型 AXIS 的当前参数不允许删除。
VAR 参数必须完整地传送。

自软件版本 SW 4.4 起:

在不完整的参数传送时可以通过系统变量 `$P_SUBPAR[i]`判别，该子程序的传送参数是否已经实际编程。

作为自变量(i)，系统变量获得传送参数的号。

系统变量 `$P_SUBPAR` 提供：

- TRUE, 如果已经编程了传送参数
- FALSE, 如果没有使用值作为传送参数。

如果说明了一个不允许的参数号，则零件程序加工中断，并发出报警。

举例：

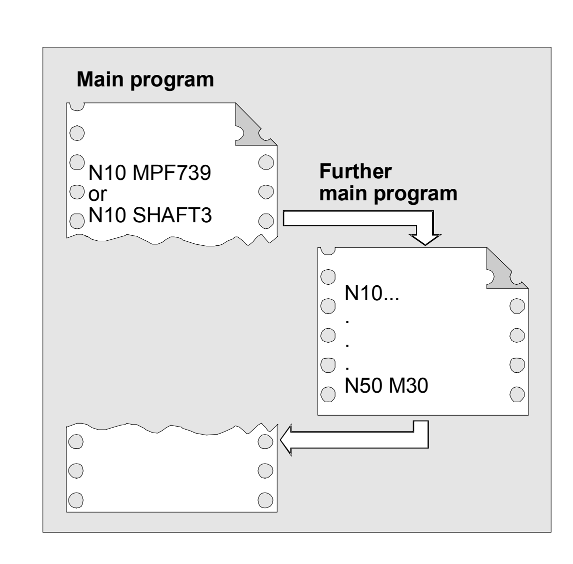
子程序

```
PROC SUB1 (INT VAR1, DOUBLE VAR2)
```

```
IF $P_SUBPAR[1]==TRUE
   ；参数 VAR1
   ；在零件程序调用中编程
ELSE
   ；参数 VAR1
   ；没有在零件程序调用中
   ；编程，并且由系统
   ；用缺省值0预置
ENDIF
IF $P_SUBPAR[2]==TRUE
   ；参数 VAR2
   ；在零件程序调用中编程
ELSE
   ；参数 VAR2
   ；没有在零件程序调用中
   ；编程，并且由系统
   ；用缺省值 0.0 预置
ENDIF
；参数3没有定义
IF $P_SUBPAR[3]==TRUE -> Alarm 17020
M17
```

调用主程序作为子程序

一个主程序也可以作为子程序调用。在主程序中设置的程序结束 M2 或者 M30 在这种情况下如同 M17（程序结束，返回到所调用的程序）处理。



通过给出程序名称编程此调用。

举例：

```
N10 MPF739 或者
N10 WELLE3
```



相应地也可以把一个子程序作为主程序启动。

2.5 可设定参数的子程序返回（自软件版本 SW 6.4 起）



编程

可设定参数的子程序返回，带重要的参数。

RET (<程序段号/标签>, <在带程序段号/标签程序段之后的程序段>, <返回级面个数>), <返回到程序开始>)	
RET (<程序段号/标签>, < >, < >)	
RET (, , <返回级面个数>, <跳回到程序开始>,	经几个级面的子程序返回（跳转返回所给定子程序级面的个数）。

说明

<程序段号/标签>	1.参数：程序段的程序段号或者作为字符串的标签（常量或者变量），程序加工应在此程序段处继续进行。 在所调用的程序中进行程序处理，用带“程序段号/标签”的程序段继续。
<在带程序段号/标签程序段之后的程序段>,	2. 类型 INTEGER 的参数 如果值大于 0，则在“程序段号/标签”之后继续下一个程序段。如果值等于 0，则子程序返回到带<程序段号/标签>的程序段。
<返回级面个数>,	3.参数，类型 INTEGER，带允许的值 1 到 11。 值 = 1: 程序在当前的级面－1继续（如同 RET 不带参数）。 值 = 2 程序在当前的程序级面－2继续，在此跳过一个级面，等等。
<跳回到程序开始>,	4. 参数，BOOL类型 **值 1 或者 0。** 值 = 1: 如果返回到主程序，并且那里有一个ISO-Dialekt-Mode有效，则分路到程序开始。

功能

通常情况下，从一个带程序结束RET或者M17的子程序返回到所调用的程序，并且零件程序的加工以在子程序调用之后的程序行继续。但是也有其它的应用情况，在此希望程序加工在另一处继续：

- 在调用切削循环 ISO-Dialekt-Mode 之后，根据轮廓描述继续程序加工。
- 在任意一个子程序级面（也在 ASUP 之后），在故障处理时返回到主程序。

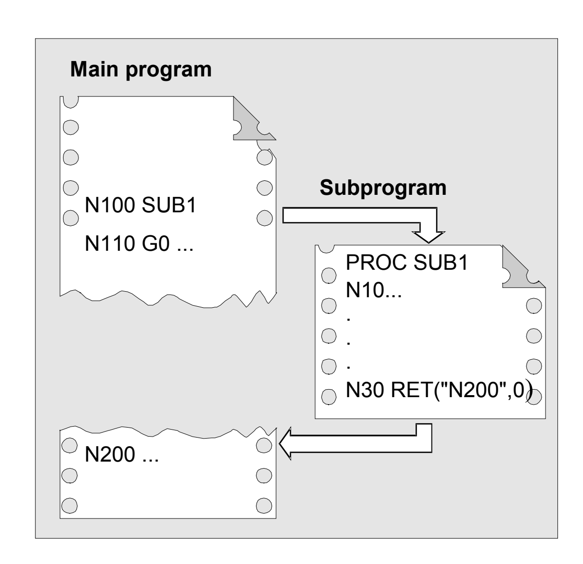
返回经过几个程序级面，用于在编译循环和 ISO-Dialekt-Mode 中的特殊应用。

使用参数化的指令 RET 可以用 4 个参数满足要求：

1. <程序段号/标签>
2. <在带程序段号/标签的程序段之后的程序段>,
3. <返回级面个数>
4. <跳回到程序开始>

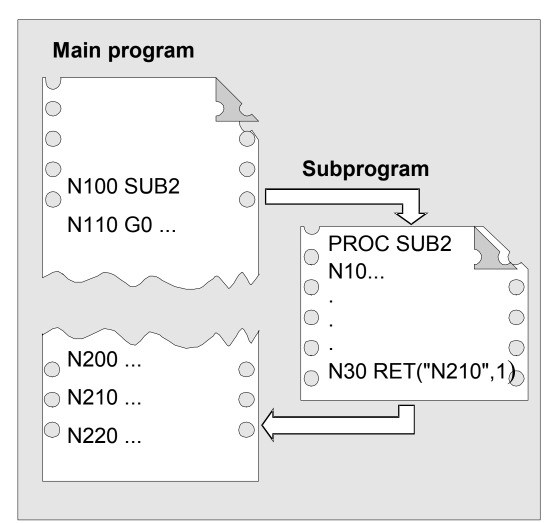
1. <程序段号/标签>

在所调用的程序中（主程序）用带“程序段号/标签”的程序段继续。



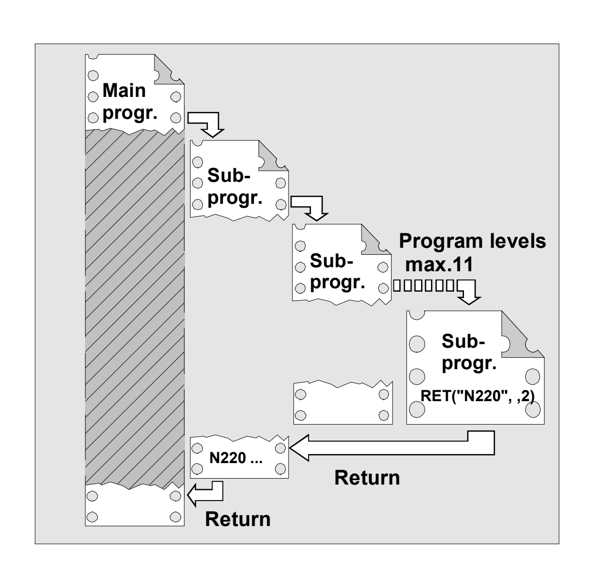
2. <在带程序段号/标签程序段之后的程序段>

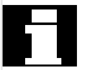
在带<程序段号/标签>的程序段之后的程序段中进行子程序返回。



3. <返回级面个数>

当前的程序级面减去<返回级面的个数>，继续程序执行。





不允许的返回级面

如果给返回级面的个数

- 编程一个负值，或者
- 编程一个值大于当前有效级面 -1（最大 11）

则用参数 5 给出报警 14091。

返回，带 SAVE 指令

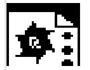
在跳回几个程序级面时，计算各个程序级面的 SAVE 指令。

在返回时模态子程序有效

如果在跳回几个程序级面时有一个模态子程序有效，并且在一个跳转的子程序中编程了撤销选择指令 MCALL用于模态子程序，则该模态子程序仍然保持有效。



用户必须始终保证在跳回几个程序级面时，以正确的模态设置继续。
比如这可以通过编程一个相应的主程序段来达到。

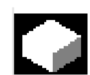


编程示例 1

故障处理：在 ASUP 加工之后，在主程序中继续。

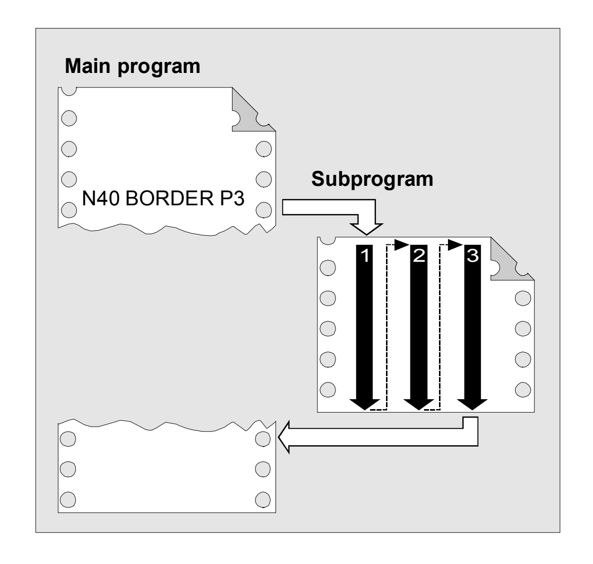
N10010 CALL "UP1"	；程序级面 0，主程序
N11000 PROC UP1	；程序级面 1
N11010 CALL "UP2"	
N12000 PROC UP2	；程序级面 2
N19000 PROC ASUP	；程序级面 2（ASUP 加工）
... RET("N10900", , ...	；程序级面 3
N19100 RET(N10900, ,$P_STACK)	；子程序返回
N10900	；在主程序中继续
N10910 MCALL	；关闭模态子程序
N10920 G0 G60 G40 M5	；修正其它的模态设置

2.6 子程序，带程序重复： P



程序重复, P

如果要求连续多次执行一个子程序，则可以在程序段中调用子程序时，在地址 P
下编程程序重复的次数。



举例：

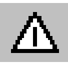
```
N40 RAHMEN P3
```

该子程序 RAHMEN 应该连续执行 3 次。

值范围

P: 1…9999

对于每个子程序调用适用于：



子程序必须位于独立的程序段中编程。

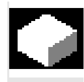
子程序调用，带重复次数和参数传送



参数仅在程序调用时或者第一次执行时传送。对于其它的重复，这些参数保持不变。

如果您在程序重复时要修改参数，则您必须在子程序中确定相应的协议。

2.7 模态子程序： MCALL



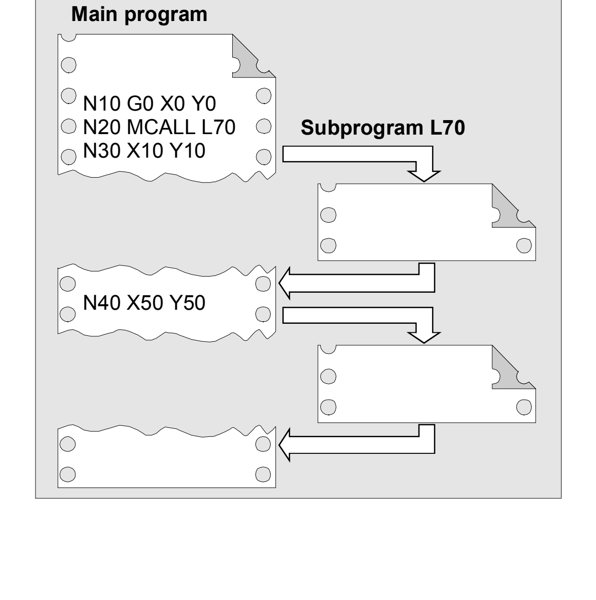
模态有效的子程序调用, MCALL

用此功能，子程序可以在每个带轨迹运行的程序段之后自动调用和执行。

为此可以自动化子程序调用，这些子程序应在不同的工件位置处执行。比如用于加工的钻孔图。

举例：

```
N10 G0 X0 Y0
N20 MCALL L70
N30 X10 Y10
N40 X50 Y50
```



在程序段 N30 和 N40
中返回编程的位置，并接着执行子程序 L70。

```
N10 G0 X0 Y0
N20 MCALL L70
N30 L80
```

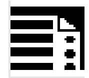
在这个例子中，在子程序 L80
中有以下带编程轨迹轴的 NC 程序段。L70 通过
L80 调用。



*在一个程序运行过程中仅可以同时有一个 MCALL
调用有效。在 MCALL 调用中仅传送一次参数。
模态子程序在下面的情况下也可以不编程一个运动而
调用：
当 G0 或 G1有效时，编程地址 S 和 F。
G0/G1单独编程在程序段中，或者与其它的 G
指令一起编程。*

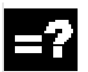
关闭模态子程序调用
使用 MCALL 不调用子程序，
或者通过编程一个新的模态子程序调用，用于一个新
的子程序。

2.8 子程序间接调用： CALL



编程

CALL <程序名>



说明

CALL	关键字用于 间接调用子程序
<程序名>	变量或者字符串类型常量 程序名，包含待执行程序部分。



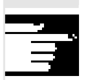
间接调用子程序, CALL

根据所给定的条件，可以在一个地点调用不同的子程序。

这里子程序名称存放在一个字符串类型的变量中。子程序调用通过CALL和变量名进行。



间接调用子程序仅可以用于没有参数传送的子程序。



为了直接调用一个子程序，存放该名称在一个字符串常量中。

举例：

直接调用字符串常量：

```
CALL "/_N_WKS_DIR/_N_SUBPROG_WPD/_N_TEIL1_SPF"
```

间接调用，通过变量：

```
DEF STRING[100] PROGNAME
PROGNAME="/_N_WKS_DIR/_N_SUBPROG_WPD/_N_TEIL1_SPF"
CALL PROGNAME
```

子程序 TEIL1 分配到变量 PROGNAME 中。使用 CALL 和路径说明，间接调用子程序。

2.9 程序部分重复，带间接编程（自软件版本 SW 6.4 起）

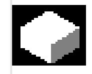
编程

```
CALL <程序名> BLOCK <起始标签> TO <结束标签>
CALL BLOCK <起始标签> TO <结束标签>
```

说明

CALL	关键字用于 间接调用子程序
<程序名>（选件）	变量或者字符串类型常量，程序名，它包含待加工的程序部分。 如果没有编程 <程序名>， 则在当前的程序中查找带<起始标签> <结束标签> 的程序部分，并且执行。
	关键字用于

BLOCK ... TO ...	间接程序部分重复
<起始标签> <结束标签>	变量或者字符串类型常量 参阅待加工程序部分的开始处或者结束处。



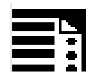
功能

使用 CALL 可以间接调用子程序，在该子程序中用 BLOCK 定义的程序部分重复按照起始标签和结束标签进行。

编程举例

```
DEF STRING[20] STARTLABEL, ENDELABEL
STARTLABEL = "LABEL_1"
ENDELABEL = "LABEL_2"
...
CALL "CONTUR_1" BLOCK STARTLABEL TO ENDELABEL ...
M17
PROC CONTUR_1 ...
LABEL_1                                  ；程序部分重复开始
N1000 G1 ...
LABEL_2                                  ；程序部分重复结束
```

2.10 用 ISO 语言编程的程序间接调用： ISOCALL



编程

ISOCALL <程序名>



说明

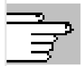
ISOCALL	子程序调用，由此激活机床数据中设定的ISO-Mode。
<程序名>	变量或者字符串类型常量 用 ISO 语言编程的程序名称。



功能

利用间接程序调用 ISOCALL，可以调用一个用 ISO 语言编程的程序。由此激活机床数据中设定的 ISO-Mode。

在程序结束处，原先的加工方式再次生效。如果在机床数据中没有设定 ISO 方式，则子程序调用以西门子方式进行。



其它的有关 ISO 方式的信息参见
/FBFA/, “ISO-语言功能说明“。

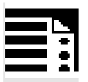
举例：

一个带循环编程的轮廓从 ISO 方式调用：

`%_N_0122_SPF`
`N1010 G1 X10 Z20`
`N1020 X30 R5`
`N1030 Z50 C10`
`N1040 X50`
`N1050 M99`
	以 ISO 方式进行的轮廓说明

`N0010 DEF STRING[5] PROGNAME = "0122"`
`...`
`N2000 R11 = $AA_IW[X]`
`N2010 ISOCALL PROGNAME`
`N2020 R10 = R10+1`
`N2300 ...`
`N2400 M30`
	西门子零件程序（循环）
	
	在 ISO 方式执行程序 0122.spf

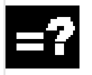
2.11 调用带路径说明和参数的子程序： PCALL



编程

带绝对的路径说明和参数传送调用子程序

```
PCALL <路径/程序名>(参数 1, …, 参数 n)
```



说明

PCALL	关键字，用于带绝对路径说明的子程序调用
<路径名>	绝对的路径说明，以“/”开始，包括子程序名。 如果没有说明绝对路径，则 PCALL 表现如同一个带程序名的标准子程序调用。 程序名说明，不带引导符 _N_ 和扩展名。 如果程序名应带引导符和扩展名编程，则它必须清楚地用引导符和扩展名说明。

参数1到n 实际参数符合子程序的 PROC 指令。



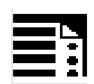
功能

利用 PCALL 可以调用带绝对路径说明和参数传送的子程序。

举例：

```
PCALL/_N_WKS_DIR/_N_WELLE_WPD/WELLE(参数1, 参数2, …)
```

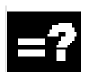
2.12 在子程序调用时用 CALLPATH 扩展查找路径（自软件版本 SW 6.4 起）



编程

除了 NCK 已经存在的文件系统之外，补充所存储的子程序，用于已经存在的 NCK 文件系统。

```
CALLPATH <路径名>
```



说明

CALLPATH 关键字，用于可编程的查找路径扩展。指令 CALLPATH 在一个自身的零件程序行中编程。

<路径名> 常量或者字符串类型的变量包含一个目录的绝对路径说明，以“/”开始，使查找路径扩展。路径必须完整地带前缀和后缀说明。
（比如：
/_N_WKS_DIR/_N_WST_WPD)
如果<路径名>包含一个空字符串，或者调用不带参数的 CALLPATH， 则查找路径指令被再次复位。最大的路径长度达到 128 个字节。



功能

使用指令 CALLPATH 可以扩展查找路径用于子程序调用。由此也可以从一个没有选择的工件目录中调用子程序，而不对子程序进行完整的、绝对的路径名称说明。在登记用户循环之前进行查找路径的扩展。(_N_CUS-DIR)。

举例：

```
CALLPATH ("/_N_WKS_DIR/_N_MYWPD_WPD")
```

因此设定以下的查找路径（位置5是新的）：

1. 当前的目录/子程序名称
2. 当前的目录/子程序名称 _SPF
3. 当前的目录/子程序名称 _MPF
4. /_N_SPF_DIR/子程序名称 _SPF
5. **/_N_WKS_DIR/_N_MYWPD/子程序名称 _SPF**
6. N_CUS_DIR/_N_MYWPD/子程序名称 _SPF
7. /_N_CMA_DIR/子程序名称 _SPF
8. /_N_CST_DIR/子程序名称 _SPF

撤销选择查找路径扩展

查找路径扩展通过以下的事件撤销选择：

- CALLPATH 带空字符串
- CALLPATH 没有参数
- 零件程序结束
- 复位

其它说明

- CALLPATH 检查所编程的路径名是否实际存在。在故障情况下，零件程序加工带补偿程序段报警 14009 中断。
- CALLPATH 也可以在INI文件中编程。它对INI文件的加工时间有影响（WPD-INI-文件或者初始化程序，用于 NC 有效的数据，比如第一通道中的框架 _N_CH1_UFR_INI）。然后初始化程序再次复位。

2.13 抑制当前的程序段显示： DISPLOF



编程

```
PROC … DISPLOF
```



功能

用 DISPLOF 抑制子程序的当前程序段显示。
DISPLOF 位于 PROC 指令的结束处。
显示循环的调用或者子程序的调用，而不显示当前的程序段。

正常情况下打开程序段显示。用 DISPLOF 关闭程序段显示，直至从子程序返回或者程序结束。如果从带 DISPLOF 属性的子程序中调用其它的子程序，则在这个子程序中也抑制当前的程序段显示。如果一个子程序带抑制的程序段显示，由一个异步的子程序中断，则当前子程序的程序段被显示。



编程举例

抑制循环中当前的程序段显示

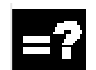
%_N_CYCLE_SPF	
;$PATH=/_N_CUS_DIR	
PROC CYCLE (AXIS TOMOV, REAL POSITION) SAVE DISPLOF	
	；抑制当前的程序段显示
	；现在循环调用作为当前的程序段显示
	；比如： CYCLE(X, 100.0)
DEF REAL DIFF	；循环内容
G01 …	；
…	
RET	；子程序返回，所调用程序后面的程序段再次显示

2.14 单段抑制：SBLOF, SBLON (自软件版本 SW 4.3 起)



编程

```
PROC ... SBLOF    ；该指令可以位于一个 PROC 程序段中，或者单独位于程序段中
SBLON             ；该指令必须位于一个独立的程序段中
```



说明

SBLOF	关闭单段
SBLON	再次接通单段



功能

程序专用的单段抑制

用 SBLOF 标记的程序，在每个单段类型时如同一个程序段完全执行。SBLOF 位于 PROC 行，并且一直有效，直至子程序结束或者中断。

使用返回指令判断在子程序结束处是否被停止。

用 **M17** 返回：
在子程序结束处停止

用 **RET** 返回：
在子程序结束处没有停止

SBLOF 也适用于所调用的子程序。

举例说明子程序，没有在单段中停止

```
PROC BEISPIEL SBLOF
G1 X10
RET
```

在程序中单段抑制

SBLOF 必须单独在程序段中。从这个程序段起，关闭单段至

- 下一个 SBLON，或者
- 有效子程序级面的结束。

举例：

```
N10 G1 X100 F1000
N20 SBLOF
N30 Y20
N40 M100
N50 R10=90
N60 SBLON
N70 M110
N80 ...
```

N20 和 N60 之间的区域，在单段运行时作为一步处理。

异步子程序单段禁止

为了在一步中执行单段的一个ASUP，必须在 ASUP 中编程一个带 SBLOF 的 PROC 指令。

这也适用于功能“可编辑的系统 ASUP”，通过 MD11610： ASUP_EDITABLE。

举例说明“可编辑的系统ASUP”：

```
N10 PROC ASUP1 SBLOF DISPLOF
N20 IF $AC_ASUP=='H200'
N30   RET                          在 BA 转换时没有重新定位
N40 ELSE
N50   REPOSA                       在所有其它情况下重新定位
N60 ENDIF
```

在单段中的程序影响

在单段功能中，用户可以按程序段方式执行零件程序。单段有以下的设定方式：

- SBL1: IPO单段，在每个加工功能程序段之后停顿
- SBL2: 单段，在每个程序段之后停顿
- SBL3: 在循环中停顿（通过选择 SBL3 抑制 SBLOF 指令）。

程序嵌套时单段抑制

如果在一个子程序中编程 SBLOF 在 PROC 指令中，则用 M17 停止到子程序返回。

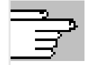
由此防止在调用的程序中已经执行下一个程序段。

如果在一个子程序（带 SBLOF 在 PROC 指令中）中激活一个单段抑制，则在调用程序的下一个加工功能程序段之后才停止。

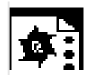
如果不希望如此，则在子程序中在返回之前（M17）
必须再次编程 SBLON。
在一个上一级的程序中，在用 RET 返回时，
不停止。

边界条件

- 当前的程序段显示可以用 DISPLOF
  在循环中抑制。
- 如果 DISPLOF 连同 SBLOF
  一起编程，则在循环之内在单段停止时，如同在
  调用循环之前一样显示。
- 如果在系统 ASUP 或者用户 ASUP
  中，单段停止用位 0＝1 或者位 1＝1（机床数据
  MD10702：IGNORE_SINGLEBLOCK_MASK ）
  抑制，则通过在 ASUP 中编程 SBLON，
  可以再次激活单段停止。
- 在用户 ASUP 中，单段停止由 MD 20117:
  IGNORE_SINGLEBLOCK_ASUP
  抑制，并且不可以通过编程 SBLON 再次激活。
- 通过选择 SBL3，抑制指令 SBLOF。
- **自软件版本 SW 6.4** 起
  在单段类型2中拒绝单段停止。
  当MD 10702: IGNORE_SINGLEBLOCK_MASK
  中位 12＝1 设置时，在单段类型 2（SBL2）
  中，在 SBLON 程序段不停止。



有关程序段显示（带/不带单段抑制）的其它信息参见
/FB/, K1 BAG, 通道, 程序运行“单段”。



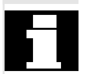
编程示例1

循环对于用户应如同一个指令作用

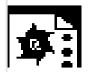
主程序

```
N10 G1 X10 G90 F200
N20 X-4 Y6
N30 CYCLE1
N40 G1 X0
N50 M30
```

```
Programm cycle:1
N100 PROC CYCLE1 DISPLOF SBLOF                    ；抑制单段
N110 R10=3*SIN(R20)+5
N120 IF (R11 <= 0)
N130 SETAL(61000)
N140 ENDIF
N150 G1 G91 Z=R10 F=R11
N160 M17
```



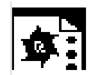
循环 CYCLE1 在单段有效时执行，也就是说在执行 CYCLE1 时必须按以下启动键。



编程示例 2

由**PLC** 启动的**ASUP**，用于激活已经改变的零点偏移和刀具补偿，不应该可见。

```
N100 PROC NV SBLOF DISPLOF
N110 CASE $P_UIFRNUM OF 0 GOTOF _G500
      -->1 GOTOF _G54 2 GOTOF _G55 3
      -->GOTOF _G56 4  GOTOF _G57
      -->DEFAULT GOTOF END
N120 _G54: G54 D=$P_TOOL T=$P_TOOLNO
N130 RET
N140 _G54: G55 D=$P_TOOL T=$P_TOOLNO
N150 RET
N160 _G56: G56 D=$P_TOOL T=$P_TOOLNO
N170 RET
N180 _G57: G57 D=$P_TOOL T=$P_TOOLNO
N190 RET
N200 END: D=$P_TOOL T=$P_TOOLNO
N210 RET
```



编程举例3

自软件版本SW6.4起，用 MD 10702:IGNORE_SINGLEBLOCK_MASK, 位 12 = 1 不再停止。

在单段类型SBL2（在每个零件程序行中停止），在SBLON指令中

; SBL2有效	
;$MN_IGNORE_SINGLEBLOCK_MASK = `H1000`	；设置 MD 10702:位 12 = 1
N10 G0 X0	；在此零件程序行中停止
N20 X10	；在此零件程序行中停止
N30 CYCLE	；由循环产生的运行程序段
PROC CYCLE SBLOF	；抑制单段停止
N100 R0 = 1	
N110 SBLON	；因为MD 10702:位 12 = 1，不再停止
N120 X1	；在此零件程序行中停止
N140 SBLOF	
N150 R0 = 2	
RET	
N50 G90 X20	；在此零件程序行中停止
M30	



编程举例4

程序嵌套时单段抑制

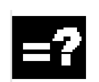
	；单段有效
N10 X0 F1000	；在此程序段中停止
N20 UP1(0)	；
PROC UP1(INT _NR) SBLOF	；单段“关”
N100 X10	；
N110 UP2(0)	
PROC UP2(INT _NR)	；
N200 X20	；
N210 SBLON	；单段“开”
N220 X22	；在此程序段中停止
N230 UP3(0)	
PROC UP3(INT _NR)	
N302 SBLOF	；单段“关”
N300 X30	
N310 SBLON	；单段“开”
N320 X32	；在此程序段中停止
N330 SBLOF	；单段“关”
N340 X34	
N350 M17	；SBLOF 有效
N240 X24	；在此程序段中停止，SBLON有效
N250 M17	；在此程序段中停止，SBLON有效
N120 X12	
N130 M17	；在此返回程序段中停止，PROC指令的SBLOF有效
N30 X0	；在此程序段中停止
N40 M30	；在此程序段中停止

2.15 执行外部子程序：EXTCALL (自软件版本 SW 4.2起)



编程

```
EXTCALL (<路径/程序名>)
```



说明

EXTCALL\	用于子程序调用的关键字

<路径/程序名> 字符串类型的常量/变量。
可以说明一个绝对的路径名或者一个程序名。
程序名说明，带/不带引导符_N_和扩展名。一个扩展名可以用符号<_>添加到程序名中。

举例：

```
EXTCALL ("/_N_WKS_DIR/_N_WELLE_WPD/_N_WELLE_SPF") 或者
EXTCALL ("WELLE")
```



功能

使用 EXTCALL
您可以由HMI后装载一个程序，方式“执行外部程序”。在此所有通过HMI的目录结构可以到达的程序可以后装载并执行。

一个外部程序路径的说明

通过 SD 42700:EXT_PROG_PATH
可以灵活地设定调用路径。SD42700包含一个路径说明，它与所编程的子程序名一起构成所调用程序的绝对路径名。

调用一个外部子程序

通过零件程序指令**EXTCALL**.调用一个外部子程序。
由
- 用EXTCALL编程的子程序和
- 设定数据SD42700: EXT_PROG_PATH

产生程序路径，用于外部子程序调用，通过以下内容的字符级联：
- SD 42700:EXT_PROG_PATH (比如 /_N_WKS_DIR/_N_WKST1_WPD)内容
- 作为分隔符的符号“/”（如果用SD 42700:EXT_PROG_PATH规定一个路径）
- 在EXTCALL所说明的子程序路径或者子程序名称。

SD 42700: EXT_PROG_PATH
由一个空格符预置。如果调用外部子程序，不作绝对路径说明，则在高级HMI中运行同样的查找路径，如同从NCK存储器中调用子程序时一样：

1. 当前的目录/子程序名称
2. 当前的目录/子程序名称_SPF
3. 当前的目录/子程序名称_MPF
4. /_N_SPF_DIR/
   子程序名称_SPF
5. /_N_CUS_DIR/
   子程序名称_SPF
6. /_N_CMA_DIR/
   子程序名称_SPF
7. /_N_CST_DIR/
   子程序名称_SPF

“当前目录”：为选择了主程序的目录。
“子程序名称”：为用EXTCALL编程的子程序名。

可设定的后装载存储器（FIFO缓存器）
在NCK中需要一个后装载存储器，用于在方式“执行外部程序”（主程序或者子程序）中加工一个程序。
后装载存储器的大小预设置为30K字节。
使用 MD 18360:MM_EXT_PROG_BUFFER_SIZE 可以设定后装载缓存器的大小。后装载缓存器的个数用 MD 18362:MM_EXT_PROG_BUFFER_NUM 设定。在方式“执行外部程序”中同时加工的所有程序（主程序或者子程序）必须每次设定一个后装载缓存器。

编程举例

1. 后装载程序位于高级HMI的局部硬盘上：

在设定数据 SD 42700:EXT_PROG_PATH
中存储了以下的路径：”/_N_WKS_DIR/_N_WST1”.
主程序_N_MAIN_MPF在工作存储器中，并且已经选择。

N10 PROC MAIN	
N20 ...	
N30 EXTCALL “SCHRUPPEN“	；调用外部子程序 ；SCHRUPPEN
N40 ...	
N50 M30	

子程序"SCHRUPPEN“（位于高级HMI目录结构
工件－>WST1中）：

```
N10 PROC SCHRUPPEN
N20 G1 F1000
N30 X=... Y=... Z=...
N40 ...
N90 M17
```

2. 待装载的程序位于网络驱动器或者HMI的ATA卡中。

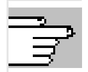
EXTCALL Windows-路径说明

调用用于网络驱动器（内置HMI或者高级HMI）。
EXTCALL \\R4711\工件\轮廓1.spf

调用用于ATA卡（内置HMI），比如
EXTCALL C:\工件\轮廓2.spf



在内置的HMI中，必须始终说明一个绝对路径。



有关操作的其它信息参见：
/BEM/ 内置HMI
/BAD/ 高级HMI



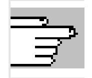
其它说明

外部子程序不允许包含跳转指令，诸如GOTOF, GOTOB, CASE, FOR, LOOP, WHILE 或者 REPEAT。

子程序调用是可以的，包括嵌套的EXTCALL调用。

自软件版本SW6.3起
IF-ELSE-ENDIF-KONSTRUKTE- 是可以的。

POWER ON, RESET
通过复位和POWER ON（上电），可以中断外部的子程序调用，并且清除各自的后装载存储器。



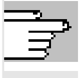
有关“执行外部程序”的其它信息参见：
/FB/, K1 BAG, 通道, 程序运行。

2.16 子程序调用，带M/T功能



功能

通过机床数据可以设定：由一个子程序调用替代T功能或者M功能。
比如这可以用于调用刀具更换程序。
在程序段查找时，带M/T功能的子程序调用性能如同标准子程序调用。



有关“带M/T功能的子程序调用”的其它信息参见：
/FB/, K1 BAG, 通道, 程序运行。

举例1： 带M6的刀具更换
M功能M6的刀具更换通过刀具更换程序WZW_UP_M6替换

N10 PROC SCHRUPPEN3	
N20 G1 F1000	
N30 X=... Y=... Z=...	
N40 T1234 M6 ;	; WZW_UP_M6 调用
M30	
相关的子程序 WZW_UP_M6:	
N110 PROC WZW_UP_M6	
...	
N130 G53 D0 G0 X=... Y=... Z=... ;	；返回刀具更换点
N140 M6 ;	；执行刀具更换
...	
N190 M17	

举例2： 带T功能编程的刀具更换：
T功能通过刀具更换程序WZW_UP_T替代

N10 PROC SCHRUPPEN4	
N20 G1 F1000	
N30 X=... Y=... Z=...	
N40 T1234 ;	; WZW_UP_T 调用
M30	
相关的子程序 WZW_UP_T:	
N310 PROC WZW_UP_T	

```
...
N330 IF $C_T_PROG == 1
N340 G53 D0 G0 X=... Y=... Z=...    ;       回到刀具更换点
N350 T=$C_T              ;                  执行刀具更换
N360 ENDIF
...
N390 M17
```

扩展T功能替代

自软件版本**SW6.4**起，T功能替代被扩展，从而通过机床数据可以设定：

是否在同时编程**D号或者DL号和T号时**

- 在一个程序段中，D或者DL根据预设置作为参数传送到T替换循环中（预设置），或者
- 应该在调用T替代循环之前执行。

通过MD 10719:T_NO_FCT_CYCLE_MODE

T功能替代的参数化用

值 0: 如同当前一样，把D或者DL号直接传送到循环，（缺省值）。

值 1: D或者DL号直接在程序段中计算。

只有设计了带M功能的刀具更换(MD 22550:TOOL_CHANGE_MODE = 1)时该功能才有效，其它情况下始终传送D或者DL值。

2.17 循环：给用户循环设定参数

文件和路径

=?

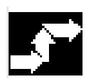
说明

cov.com	循环概述
uc.com	循环调用说明



功能

用这些文件可以给自身的循环设定参数。



操作顺序

文件cov.com以标准循环提供，并且相应地扩展。文件uc.com由用户自己编制。

两个文件必须装载到非激活文件系统的目录“用户循环”下（或者在程序中以相应的路径说明：;$PATH=/_N_CUS_DIR）。

匹配 cov.com – 循环概述

以标准循环提供的文件cov.com有以下的结构：

%_N_COV_COM	文件名
;$PATH=/_N_CST_DIR	路径说明
;Vxxx 11.12.95 Sca 循环概述	注释行
C1(CYCLE81) 钻孔, 对中	调用用于第一个循环
C2(CYCLE82) 钻孔, 锪平面	调用用于第二个循环
...	
C24(CYCLE98) 螺纹链	调用用于最后一个循环
M17	文件结束

对于每个新来的循环，需要在下面的句法中添加一行：

C<号> (<循环名>) 注释文本

序号：一个任意整数，到目前为止在该文件中还不允许使用；

循环名：待捆绑循环的程序名

注释文本：可以选择，用于循环的一个注释文本

举例：

C25 (MEIN_ZYKLUS_1) 用户循环_1

C26 (特殊循环)

举例说明文件uc.com -

用户循环说明

利用示例说明：

对于后面的两个循环，应该重新编制一个循环参数设定：

PROC MEIN_ZYKLUS_1 (REAL PAR1, INT PAR2, CHAR PAR3, STRING[10] PAR4)	
;循环有以下的传送参数：	
;	
;PAR1:	实际值，范围 -1000.001 <= PAR2 <= 123.456, 用 100 预置
;PAR2:	正的整数值，范围 0 <= PAR3 <= 999999, 用 0 预置
;PAR3:	1个ASCII-字符
;PAR4:	长度10的字符串，用于一个子程序名
;	
...	
M17	

PROC SPEZIALZYKLUS (REAL WERT1, INT WERT2)	
;循环有以下的传送参数：	
;	
;WERT1:	实际值，没有值范围限制和预置
;WERT2:	整数值，没有值范围限制和预置
...	
M17	

有关的文件uc.com

```
%_N_UC_COM
;$PATH=/_N_CUS_DIR
//C25(MEIN_ZYKLUS_1) 用户循环_1
(R/-1000.001 123.456 / 100 /该循环的参数_2)
(I/0 999999 / 1 / 整数值)
(C//"A" / 符号参数)
(S///子程序名)

//C26(SPEZIALZYKLUS)
(R///总长度)
(I/*123456/3/加工方式)
M17
```

句法说明，用于文件uc.com-
用户循环说明

每个循环的开始行：
如同在文件cov.com中一样，带前置的"//"

```
//C <号> (<循环名>) 注释文本
```

举例：

```
//C25(MEIN_ZYKLUS_1) 用户循环_
```

每个参数的说明行：

(<数据类型标记> / <最小值> <最大值> / <预置值> /
<注释>)

数据类型标记：

R	用于实数
I	用于整数
C	用于字符（1个字符）
S	用于字符串

最小值，最大值（可以取消）
待输入值的极限，在输入时检测；超出该范围的值不
可以输入。

可以说明计数值，它们可以用触发键操作；这些值以
“*”开始计数，其它的值不允许。

举例：
(I/*123456/1/加工方式)
在字符串类型和字符类型时没有极限；

预置值（可以取消）
该值在调用循环时在相应的表征码中预置；它可以通
过操作修改。

注释
文本，最多50个字符，在用于循环的调用表征码中、
在该参数输入数组之前显示。

显示示例，用于两个循环

显示表征码，用于循环 MEIN_ZYKLUS_1

Parameter 2 of the cycle	100
Integer value	1
Character parameter	
Subprograms	

显示表征码，用于循环SPEZIALZYKLUS

Total length	100
Type of machining	1

2.18 宏指令技术 DEFINE...AS

什么是宏指令？

作为宏指令，是指单个的指令组合成一个新的总指令，带自己的名称。G-,M-和H-功能或者L-子程序名也可以作为宏指令编制。
在程序运行中调用该宏指令时，可以在该宏指令名下一个接一个地执行编程的指令。

宏指令使用

总是反复的指令序列，人们仅编程一次，在一个自身的宏指令模块中作为宏指令，或者仅在程序开始处出现一次。
该宏指令可以在任意一个主程序中或者子程序中调用并执行。

编程

宏指令用关键字DEFINE…AS标识。

宏指令定义为：

```
DEFINE  NAME AS  <指令>
```

举例：

宏指令定义：

```
DEFINE LINIE AS G1 G94 F300
```

在NC程序中调用：

```
N20  LINIE X10 Y20
```

激活宏指令

- 在软件版本 **SW 4** 之前

  在POWER ON（上电）之后一个宏指令有效。

- 自软件版本**SW5**起

  当宏指令装载到NC时（软键“装载”）它才有效。

三位的 M-/G-功能 (自 SW 5起)

- 在软件版本 **SW 4** 之前

  在编程一个三位的M功能之后，给出报警12530。

- 自软件版本 **SW 5**起

  可以编程三位的M功能和G功能。

  举例：

  N20 DEFINE M100 AS M6

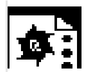
  N80 DEFINE M999 AS M6



其它说明

不可以嵌套宏指令。

H功能和L功能可以两位编程。



编程举例

宏指令定义举例

DEFINE M6 AS L6	在刀具更换时调用一个子程序，该子程序接收了必需的数据传送。在子程序中输出自身的刀具更换M功能（比如M106）。
DEFINE G81 AS DRILL(81)	模仿DIN-G功能
DEFINE G33 AS M333 G333	在切削螺纹时要求与PLC的同步。原来的G功能G33通过MD改名为G333，对于用户来说编程保持不变。

举例说明一个全局宏指令文件：

在控制系统中读入该宏指令文件之后，激活宏指令（参见上面）。现在可以在零件程序中使用这些宏指令。

%_N_UMAC_DEF	
;$PATH=/_N_DEF_DIR ；用户专用的宏指令	
DEFINE PI AS 3.14	
DEFINE TC1 AS M3 S1000	
DEFINE M13 AS M3 M7	；主轴右转，冷却液开
DEFINE M14 AS M4 M7	；主轴左转，冷却液开
DEFINE M15 AS M5 M9	；主轴停止，冷却液关
DEFINE M6 AS L6	；调用刀具更换程序
DEFINE G80 AS MCALL	；撤销选择钻削循环
M30	；



- *关键字和所保留的名称不允许用宏指令过定义。*
- *使用宏指令技术可以很大地改变系统的编程语言！因此您必须要特别小心地使用宏指令技术！*
- *宏指令也可以在NC程序中约定。只有命名符才允许用作宏指令名称。G功能宏指令仅可以在宏指令模块中由系统全局约定。*
- *使用宏指令技术可以定义任意的命名符、G-/M-/H-功能和L-程序名。*
- *允许宏指令名带1个字母和1个数字（仅在FM-NC）。*

■

用于记录

文件和程序管理

3.1 概述

存储器结构

有一个存储器结构供用户使用，它分为两个部分。

1. 工作存储器

工作存储器包含当前的系统数据和用户数据，控制系统以此数据运行（有源文件系统）。

举例：

有效的机床数据，刀具补偿数据，零点偏移。

2. 程序存储器

在程序存储器中存储文件和程序，并且可以长期保存（无源文件系统）

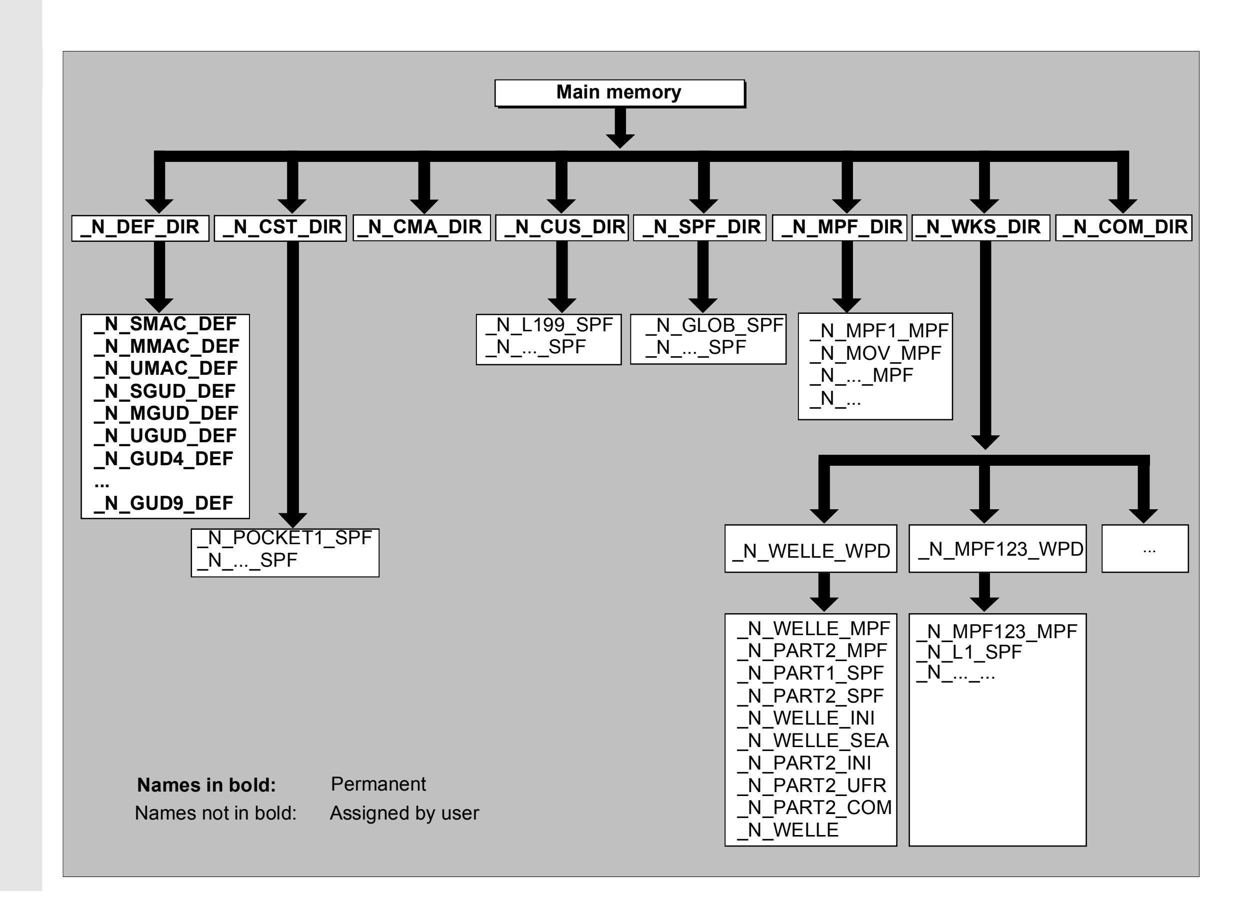
举例：

主程序和子程序，宏指令定义。

3.2 程序存储器

概述

主程序和子程序存储在零件存储器中。除此之外还有一些文件类型可以中间存储，在需要时（比如加工某一个工件）传送到工作存储器中（比如用于初始化目的）。



目录

正常情况下有以下目录：

1. _N_DEF_DIR	数据块和宏指令块
2. _N_CST_DIR	标准循环
3. _N_CMA_DIR	机床制造商循环
4. _N_CUS_DIR	用户循环
5. _N_WKS_DIR	工件
6. _N_SPF_DIR	全局子程序
7. _N_MPF_DIR	主程序标准目录
8. _N_COM_DIR	注释标准目录

文件类型

在程序存储器中可以有以下文件类型：

*name*_MPF	主程序
*name*_SPF	子程序

*name*_TEA	机床数据
*name*_SEA	设定数据
*name*_TOA	刀具补偿
*name*_UFR	零点偏移/框架
*name*_INI	初始化文件
*name*_GUD	全局用户数据
*name*_RPA	R参数
*name*_COM	注释
*name*_DEF	全局用户数据和宏指令定义

工件目录, _N_WKS_DIR

工件目录在正常情况下建立在程序存储器的
_N_WKS_DIR名称下。

工件目录包含所有编程工件的相应工件目录。

工件目录, WPD标志

为了可以灵活处理数据和程序，可以把某些数据和程
序打包，或者存放在单独的工件目录下。
一个工件目录包含加工该工件所需要的所有文件。

它可以是主程序，子程序，任意初始化程序和注释文
件。

初始化程序仅执行一次，它是根据机床数据
MD 11280:WPD_INI_MODE
中的设定，在选择程序之后以第一个零件程序起始。

举例：
为工件WELLE所编制的工件目录_N_WELLE_WPD包含
以下的文件：

_N_WELLE_MPF	主程序
_N_PART2_MPF	主程序
_N_PART1_SPF	子程序
_N_PART2_SPF	子程序
_N_WELLE_INI	用于此工件文件的共同的初始化程序
_N_WELLE_SEA	设定数据初始化程序
_N_PART2_INI	用于程序第2部分文件的共同的初始化程序
_N_PART2_UFR	用于程序第2部分框架文件的初始化程序
_N_WELLE_COM	注释文件

在外部PC中编制工件目录



下面介绍如何在一个外部数据站中编制工件目录。

如何直接在控制系统中管理文件和程序（由PC到控制系统）参见操作说明。

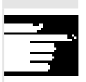
;$PATH-指令

在一个文件的第二行用$PATH=… 说明目标路径。

举例：

```
;$PATH=/_N_WKS_DIR/_N_WELLE_WPD
```

文件存放在所说明的路径下。



重要

如果不对路径进行说明，则文件类型为SPF的文件存放在/_N_SPF_DIR下，结尾为_INI的文件存放在工作存储器中，所有其它的文件则放于/_N_MPF_DIR中。

对前面示例WELLE的路径说明：

_/N_WELLE_MPF 存放在 /_N_WKS_DIR/_N_WELLE_WPD中

```
%_N_WELLE_MPF
;$PATH=/_N_WKS_DIR/_N_WELLE_WPD
N10 G0 X… Z…
•
M2

WELLE:      _/N_WELLE_SPF 存放在
            /_N_SPF_DIR中
•
%_N_WELLE_SPF
•
M17
```

选择加工工件

可以为一个通道中的加工选择一个工件目录。
如果在此目录中只有一个有此名字的主程序，或者只有一个主程序(MPF),则自动选择该主程序加工。

举例：
工件目录
/_N_WKS_DIR/_N_WELLE_WPD 包含文件
_N_WELLE_SPF 和 _N_WELLE_MPF。

自软件版本SW5起（仅MMC102/103）：
参见“操作说明”/BA/ 章节“工作表“ 以及
“选择加工程序“。

在子程序调用中查找路径

如果在零件程序中没有明确说明所调用子程序的路径（或者初始化文件），则按照一个固定的查找路径求算所调用的程序。

举例说明用绝对路径参数调用一个子程序。

```
CALL"/_N_CST_DIR/_N_CYCLE1_SPF"
```

在正常情况下不用说明路径调用程序：

举例：

```
CYCLE1
```

查找路径顺序

1. 当前的路径/名称	工件目录或者 标准目录_N_MPF_DIR
2. 当前的路径 / *名称_SPF*	
3. 当前的路径 / *名称_MPF*	
4. /_N_SPF_DIR / *名称_SPF*	全局子程序
5. /_N_CUS_DIR / *名称_SPF*	用户循环
6. /_N_CMA_DIR / *名称_SPF*	机床制造商循环
7. /_N_CST_DIR / *名称_SPF*	标准循环

在调用零件程序时编程查找路径（自软件版本SW6.4起）。

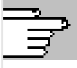
CALLPATH-指令

在调用子程序时可以使用零件程序指令CALLPATH扩展查找路径。

举例：

```
CALLPATH ("/_N_WKS_DIR/_N_MYWPD_WPD")
```

根据所说明的编程，查找路径存放在位置5之前（用户循环）。



有关用CALLPATH调用子程序时可编程的查找路径的详细情况参见章节2.12。

3.3 工作存储器

初始化程序

这里讨论工作存储器数据可以预置（初始化）时如何编程。

可以使用以下的文件类型：

*name*_TEA	机床数据
*name*_SEA	设定数据
*name*_TOA	刀具补偿
*name*_UFR	零点偏移/框架
*name*_INI	初始化文件
*name*_GUD	全局用户数据
*name*_RPA	R参数

数据区

数据可以划分为不同的区。举例来说一个控制系统一般有几个通道（不包括810D CCU1，840D NCU 571）或者通常也有几个轴。有：

标志	数据区
NCK	NCK专用数据
CHn	通道专用数据 （N说明通道号）
AXn	轴专用数据（n说明加工轴序号）。
TO	刀具数据
COMPLETE	所有数据

在外部PC中产生初始化程序

利用数据区标志和数据类型标志，可以确定数据保护时视作数组的数据区。

举例：

_N_AX5_TEA_INI	用于第5轴的机床数据
_N_CH2_UFR_INI	通道2框架
_N_COMPLETE_TEA_INI	所有机床数据

在系统开机调试之后在工作存储器中有一个数据组，它保证控制系统正常运行。

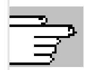
保护初始化程序

工作存储器的文件可以保护到一个外部PC中，并可以从那儿再次读入。

- 文件用 COMPLETE 保护。
- 使用INITIAL 产生一个
  INI-文件：_N_INITIAL_INI。

装载初始化程序

如果INI程序仅使用一个通道的数据，则它也可以作为零件程序选择并调用。因此也就可以初始化程序控制的文件。



所有文件类型说明均可参见操作编程说明。

多个通道的控制系统中如何进行

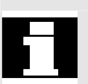
CHANDATA （通道号）用于多个
通道，仅可以在文件N_INITIAL_INI中使用。
N_INITIAL_INI是开机调试文件，用此文件可以初
始化控制系统的所有文件。

举例：

```
%_N_INITIAL_INI
CHANDATA(1)
；加工轴分配通道1
$MC_AXCONF_MACHAX_USED[0]=1
$MC_AXCONF_MACHAX_USED[1]=2
$MC_AXCONF_MACHAX_USED[2]=3
CHANDATA(2)
；加工轴分配通道2
$MC_AXCONF_MACHAX_USED[0]=4
$MC_AXCONF_MACHAX_USED[1]=5
CHANDATA(1)
；轴向机床数据
；粗准停窗口：
$MA_STOP_LIMIT_COARSE[AX1]=0.2  ；轴 1
$MA_STOP_LIMIT_COARSE[AX2]=0.2  ；轴 2
精准停窗口：
$MA_STOP_LIMIT_COARSE[AX1]=0.01  ；轴 1
$MA_STOP_LIMIT_COARSE[AX1]=0.01  ；轴 2
```



*在零件程序中，CHANDATA-
指令仅为处理NC程序的通道设定；也就是说用此指
令可以防止NC程序在一个没有事先规定的通道中处
理。
在故障时停止程序执行。*



说明

在工作表中的INI文件不含CHANDATA指令。

3.4 定义用户数据



功能



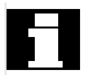
在开机调试时 (GUD)定义用户数据。
所需要的机床数据必须做相应的设置。
必须配置用户存储器。所有相关的机床数据均有名称组成GUD。

* 自软件版本**SW5**(01.99)起：
  在HMI操作界面的操作区通讯中可以定义用户数据(GUD)，从而省去费时的重新进行数据保护(%_N_INITIAL_INI)。
  适用于：
  * 在硬盘上的定义文件无效。
  * 在NC上的定义文件始终有效。

保留的模块名称

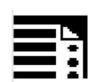
在目录/_N_DEF_DIR中可以存放以下模块：

_N_SMAC_DEF	包含宏指令定义（西门子－系统应用）
_N_MMAC_DEF	包含宏指令定义（机床制造商）
_N_UMAC_DEF	包含宏指令定义（用户）
_N_SGUD_DEF	包含全局数据定义（西门子－系统应用）
_N_MGUD_DEF	包含全局数据定义（机床制造商）
_N_UGUD_DEF	包含全局数据定义（用户）
_N_GUD4_DEF	可以自由定义
_N_GUD5_DEF	包含测量循环定义（西门子－系统应用）
_N_GUD6_DEF	包含测量循环定义（西门子－系统应用）
_N_GUD7_DEF	包含标准循环定义（西门子－系统应用）
_N_GUD8_DEF	可以自由定义
_N_GUD9_DEF	可以自由定义



说明

如果没有测量循环/标准循环，则这里保留的模块也可以自由定义。



编程

单独的GUD变量用DEF指令编程：

DEF 区 *VL-停止类型名称[.., ...]=值*



说明

区	区标记该变量作为GUD变量，并确定应用范围：	
	NCK	NCK范围
	CHAN	通道范围
VL-停止	进刀停止选件特性：	
	SYNR	在读入时进刀停止
	SYNW	在写入时进刀停止
	SYNRW	在读/写时进刀停止
类型	数据类型 BOOL REAL INT AXIS FRAME STRING CHAR	
名称	变量名	
[.., ...]	数组变量可选运行界限	
值	可选预置值， 数组中几个值，通过逗号分开，或者 REP (w1) , SET(w1, W2, ...), (w1, w2, ...) 在框架类型时不可以有初始值	

工作流程

定义用户数据（**GUD**）

1. 保护模块_N_INITIAL_INI
2. 给用户数据建立定义文件。
   - 在外部的PC中（至软件版本**SW4**）
   - 在操作区通讯中（自软件版本**SW5**起）。
3. 把定义文件装载到控制系统的程序存储器中。
4. 激活定义文件。
5. 数据保护。

2. 给用户数据建立定义文件。

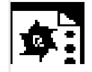
定义文件可以在外部PC中建立，也可以在操作区通讯中完成。也存在预定义的文件名（参见“预留模块名称”）：

```
_N_SGUD_DEF
_N_MGUD_DEF
_N_UGUD_DEF
_N_GUD4_DEF … _N_GUD9_DEF
```

带这些名称的文件可以包含GUD变量的定义。

3. 把定义文件装载到控制系统的程序存储器中。

控制系统总是按照缺省方式建立一个目录_N_DEF_DIR。
该名称作为路径登记到GUD定义文件头，并在由RS232接口读入时处理。



编程举例

举例说明一个定义文件，全局文件（西门子）：

%_N_SGUD_DEF	
;$PATH=/_N_DEF_DIR	
DEF NCK REAL RTP	；退回平面
DEF CHAN INT SDIS	；安全距离
M30	

4. 激活定义文件

- 至软件版本**SW4** 在读入

  _N_INITIAL_INI之前，保护所有的程序、框架和机床数据，因为静态存储器被格式化。
  定义文件只有在读入_N_INITIAL_INI-文件之后才有效。
- 自软件版本**SW5**起
  当GUD定义文件装载到NC时（软件“装载”）它才有效。参见“自动激活…”

5. 数据保护

在从工作存储器中激活文件_N_COMPLETE_GUD时，只保护文件内容。为全局用户变量所编制的定义文件必须分开存档。

全局用户数据的变量指令也存储到_N_INITIAL_INI中，名称必须与定义文件中的名称相一致。

举例说明一个定义文件，全局文件（西门子）：

%_N_MGUD_DEF	
;$PATH=/_N_DEF_DIR	
；机床制造商的全局文件定义	
DEF NCK SYNRW INT STUECKZAHL	；在读/写时包含进刀停止；控制系统中存在专门的数据 ；读写所有通道
DEF CHAN INT WERKZEUGTABELLE[100]	；用于通道专用图像的刀具表 ；刀库位置上的刀具号
M30	；给每个通道分开编制表



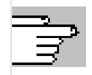
其它说明

可设计的参数范围，自软件版本SW7.1起

通过机床数据可以扩展各个GUD模块至通道专用的参数范围，它们现在也可以在同步动作中执行。由此GUD参数性能就像R参数。

新编制的GUD区可以定义数据类型REAL、INT 和BOOL。参见预定义变量名称表。

预定义变量名称表			
同步动作参数名称			
数据类型**REAL**	数据类型**INT**	数据类型**BOOL**	模块
SYP_RS[ ]	SYP_IS[ ]	SYP_BS[ ]	SGUD-模块
SYP_RM[ ]	SYP_IM[ ]	SYP_BM[ ]	MGUD-模块
SYP_RU[ ]	SYP_IU[ ]	SYP_BU[ ]	UGUD-模块
SYP_R4[ ]	SYP_I4[ ]	SYP_B4[ ]	GUD4-模块
SYP_R5[ ]	SYP_I5[ ]	SYP_B5[ ]	GUD5-模块
SYP_R6[ ]	SYP_I6[ ]	SYP_B6[ ]	GUD6-模块
SYP_R7[ ]	SYP_I7[ ]	SYP_B7[ ]	GUD7-模块
SYP_R8[ ]	SYP_I8[ ]	SYP_B8[ ]	GUD8-模块
SYP_R9[ ]	SYP_I9[ ]	SYP_B9[ ]	GUD9-模块



其它信息参见：
/IAD/,第6章 "控制系统设定参数“

3.5 定义用户数据 (GUD) 保护级

编程

模块保护级在模块头后给出

```
%_N_MGUD_DEF                  ; 模块种类
;$PATH=/_N_DEF_DIR            ; 路径
APR Wert APW n                ; 保护级在独立行中
```

说明

保护级： APW n APR n	读写保护　（读写保护） 用于写　（写） 用于读　（读）
n	保护级n 从0或者10 （最高级） 到7或者17 （最低级）

保护级n的含义：

0	或者	10	西门子
1	或者	11	OEM_高
2	或者	12	OEM_低
3	或者	13	最终用户
4	或者	14	钥匙开关3
...			...
7	或者	17	钥匙开关0

APW 0-7, APR 0-7
在MDA方式下，不可以通过NC程序读/写模块变量。

允许这些值在GUD模块中，以及用于REDEF指令中各个变量的保护级参数说明。

APW 10-17, APR 10-17:
在MDA方式下，可以通过NC程序读/写模块变量。

这些值只允许在模块专用的GUD保护级中。



说明

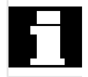
如果保护一个完整的文件，必须使这些指令位于文件的第一个定义之前，否则必须在所涉及到文件的**REDEF**指令中。



功能

通过定义读写准则，使GUD模块免受处置。在循环中GUD变量可以被询问，它们不会通过HMI操作修改，也不会由程序进行修改。
读写保护与所有在此模块中定义的变量有关。
如果进行了不允许的存取，则控制系统会给出一个报警。

第一次激活一个GUD定义文件在第一次激活一个GUD定义文件时，处理也许包含在其中的、定义的存取权，并且自动送回到GUD定义文件的读写权。



说明
在GUD定义文件中登记存取权，可以把存取权限制在GUD定义文件中，但不可进行扩展。

举例
在定义文件 _N_GUD7_DEF中： APW2
a) 文件_N_GUD7_DEF有值3作为写保护。然后值3用值2改写。
b) 文件_N_GUD7_DEF有值0作为写保护。
没有改变。

使用指令APW，文件的写权限反作用受到影响。
使用指令APR，文件的读权限反作用受到影响。

说明
在GUD定义文件中，如果人们在无意当中登记了一个较高的权限（高于自身存取），则存档文件必须重新录入。



工作流程

在第一次定义一个变量之前，在GUD模块中按照所要
求的保护级编程存取保护。
关键字必须位于一个独立的程序段中。

举例说明在带存取保护的定义文件中写
（机床制造商），读（钥匙开关2）：

编程举例

%_N_GUD6_DEF	
;$PATH=/_N_DEF_DIR	
APR 15 APW 12	；保护级，用于所有跟随的变量
DEF CHAN REAL_CORRVAL	
DEF NCK INT MYCOUNT	
…	
M30	;

3.6 自动激活GUDs和MACs（自软件版本SW4.4起）



功能

编辑用于GUD和宏指令定义的定义文件，用于
- MMC102/103的操作区通讯中。

如果在NC中编辑一个定义文件，则在离开编辑器时
出现一个询问：这些定义是否应该被设置为有效？

举例：

“您要从文件GUD7.DEF中激活这些定义吗？”

"OK" → 出现一个询问：是否应该保护当前有效的文件？

“这些定义的当前文件应该保持不变吗？”

"OK" → 保护待执行定义文件的GUD模块，激活新的定义，所保护的文件再次录入。

"停止" → 新的定义激活，旧的文件失去。

"停止" → 定义文件中的修改被拒绝，相应的数据模块没有改变。

卸载

如果卸载一个定义文件，则在显现一个询问之后清除相应的数据块。

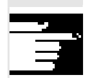
装载

如果装载一个定义文件，则出现一个询问：文件是否应该激活？或者这些文件是否应该保持不变？如果放弃激活，则文件不装载。

如果光标位于一个装载的定义文件上，则软键名称从“装载”变为“激活”，以便使该定义生效。如果选择“激活”，则再次出现一个询问：这些文件是否应该保持不变？



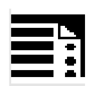
仅在变量定义文件中保护数据，在宏指令中没有。



其它说明（MMC103）

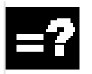
如果没有足够的存储器用于激活定义文件，则在改变存储器大小之后，文件必须从NC装载到MMC，然后再返回到NC中并激活。

3.7 改变机床数据和设定数据的保护级



编程

REDEF机床数据/设定数据保护级



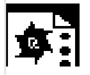
说明

REDEF	新定义 再定义
机床数据/设定数据	被赋予一个保护级的机床数据或者设定数据。
保护级： APW n APR n	读写保护 （读写保护） 用于写 （写） 用于读 （读）
n	保护级n 从0 （最高级） 到7 （最低级）



功能

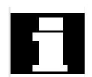
用户可以修改保护级。在机床数据中仅可以分配优先级较低的保护级，在设定数据中则要高一些。
使用口令字，使用户可以改变定义。在GUD定义文件中改变定义，比如_N_UGUD_DEF。



编程举例

在单个的MD中改变权限

```
%_N_SGUD_DEF
;$PATH=/_N_DEF_DIR
REDEF $MA_CTRLOUT_SEGMENT_NR APR 2 APW 2
REDEF $MA_ENC_SEGMENT_NR APR 2 APW 2
REDEF $SN_JOG_CONT_MODE_LEVELTRIGGRD APR 2 APW 2
M30
```

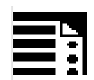


修改保护级



再次重新设置机床数据/设定数据

如果要求可以再次取消保护级的修改，则原来的保护级必须再次写回。



编程举例

在单个的MD中复位权限到原来的值。

```
%_N_SGUD_DEF
;$PATH=/_N_DEF_DIR
REDEF $MA_CTRLOUT_SEGMENT_NR APR 7 APW 2
REDEF $MA_ENC_SEGMENT_NR APR 0 APW 0
REDEF $SN_JOG_CONT_MODE_LEVELTRIGGRD APR 7 APW 7
M30
```

3.8 NC语言指令保护级 (自软件版本 SW 7.1起)

编程

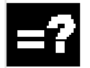
REDEF (NC-语言单元) ***APX*** *值*

REDEF (系统变量) ***APW*** *值*

REDEF (机床数据/设定数据) ***APR*** *值*

REDEF (机床数据/设定数据) ***APW*** *值*

REDEF (机床数据/设定数据) ***APR*** *值* ***APW*** *值*



说明

NC语言单元	语言单元，该语言单元应该分配一个保护级用于执行该指令： 1. 预定义的子程序/功能（同名清单） 2. 关键字“DO” （“DO”指令用于同步动作） 3. G功能（G功能/位移条件清单） 4. 用于循环的程序名称该循环必须存放在一个循环的目录下，并且包含一个PROC指令。

系统变量	系统变量，该系统变量应分配一个用于写存取的保护级（参见系统变量清单）。
机床数据/设定数据	机床数据或者设定数据，它们应分配一个读/写存取的保护级。
APX APW, APR	执行存取保护的关键字，写/读
值	保护级的数值（0到7）， 从0（最高级） 到7 （最低级）
值7	钥匙开关位置0对应所有零件程序指令的预设定。



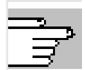
功能

用于读写机床数据/设定数据和GUD的保护级方案扩展到上面所说的零件程序指令。对此，可以用REDEF指令给一个零件程序分配一个保护级0到7。只有具备相应的执行权限后，才可以在零件程序处理中执行该指令。

REDEF指令的作用和应用

REDEF指令作用于所有通道和BAG，可以应用于：

- 预定义的子程序和功能（预定义的子程序）
- G代码，根据G功能/准备功能的清单。
- 用于写存取零件程序或者系统变量同步动作。
  始终可以读存取。
- “DO”指令，用于同步动作。
- 用于修改机床数据和设定数据的写存取或者读存取，如同现在一样。
- 用于保护循环调用。



其它信息参见：
/BAD/, HMI 操作说明, 第2章
/IAD/,开机调试，控制系统设定参数，章节“保护级方案”。



工作流程

与GUD定义类似，有自身的定义文件供使用，它们在系统引导时处理：

最终用户：　/_N_DEF_DIR/_N_UACCESS_DEF
制造商：　/_N_DEF_DIR/_N_MACCESS_DEF
西门子：　/_N_DEF_DIR/_N_SACCESS_DEF

定义文件中的子程序调用

在上面所述的定义文件中也可以调用包含REDEF指令的子程序。
这些指令如同DEF指令一样，也必须位于文件部分的开始。
子程序必须有扩展名SPF或者MPF，并且继承用$MN_ACCESS_WRITE_xACCESS设定的定义文件的写保护。

举例：

```
N10 REDEF GEOAX APX 3
N20 IF(ISFILE("/_N_CST_DIR/_N_SACCESS_SUB1_SPF"))
N30         PCALL /_N_CST_DIR/_N_SACCESS_SUB1_SPF
N40 ENDIF
N40 M17
```



REDEF指令说明，自软件版本SW7.1起

一旦用于NC语言指令的保护级功能生效，所有通过机床数据/设定数据设置的保护级（在GUD定义文件中带REDEF指令）必须按照新的定义文件修改。因此，仅在上面所述的保护级定义文件中允许设置机床数据和设定数据的保护级。

到软件版本**SW6.4**为止，原来用REDEF指令可以存取任意机床数据，而与所有定义文件中当前的存取权无关，但是现在不行，并且会发出报警15420。为此保护级0到3（西门子，机床制造商或者最终用户）有自身的定义文件供使用。



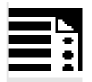
其它说明

仅在GUD定义文件中可以设置初始化属性和同步动作属性。

系统变量保护级

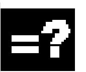
系统变量的保护级仅适用于通过零件程序指令的赋值语句。在操作界面中，各个HMI高级的/内置的保护级方案生效。

3.9 修改NC语言单元的属性（自软件版本SW6.4起）



编程

```
REDEF NC-语言单元属性属性值
```



说明

NC语言单元	以下适用： GUD R参数 机床数据/设定数据 同步变量 ($AC_PARAM, $AC_MARKER, $AC_TIMER) 来自零件程序可写的系统变量（参见附录） 用户框架（G500，等等） 刀库/刀具配置	
属性		允许：
	INIPO	GUD, R-参数, 同步变量
	INIRE	" " "
	INICF	" " "
	PRLOC	设定数据
	SYNR	GUD
	SYNW	"
	SYNRW	"
	APW	机床数据/设定数据
	APR	" " "

值	在属性INIPO、INIRE、INICF和PRLOC中可选的参数： 后来的起始值 格式： 单个值　　比如5 数值清单　　比如(0, 1, 2, 3, 4, 5, 6, 7, 8, 9) 　　用于变量，带10个单元 `REP` (w1)　　带w1:待重复的 　　值清单，用于带几个单元的变量 　　比如REP(12) `SET(w1, w2, w3, ...)`或者 `(w1, w2, w3, ...)`值清单 `n`　　所要求的参数保护级，当属性为 　　APR或者APW时 对于**GUD**在定义时可以说明一个起始值(DEF NCK INT _MYGUD=5)。如果该起始值（比如在DEF NCK INT _MYINT时）没有说明，则在REDEF指令中可以后来再确定该起始值。 不可以用于R参数和系统变量。 仅可以赋值常量。表达式不允许作为数值。



功能

前面章节中所描述的功能（定义文件目标和确定保护级）在软件版本SW6.4以后，使用REDEF指令作为一个公共的接口，用于设定属性和数值。

属性的含义

INIPO

<u>**INI**</u>t 在开机上电时
再次启动NC时，数据被缺省值改写。

INIRE

<u>**INI**</u>t 在操作面板前方复位或者TP结束
在主程序结束时，比如M2,M30等等，或者用复位键停止时，数据由缺省值覆盖。
在INIPO时INIRE也生效。

INICF	**INI**t 在**N**ew**C**onf请求时或者TP指令NEWCONF时 在NewConf请求时或者TP指令NEWCONF时，数据以缺省值覆盖。 在INIRE 和INIPO时INICF也生效。

用户必须实现初始化事件的同步，也就是说如果一个零件程序在两个不同的通道中结束，则在每个过程中对变量进行初始化。这对全局数据或者轴向数据生效！

给定一个缺省值：

如果用
REDEF <名称> INIRE, INIPO; INICF; PRLOC
改变一个系统变量的特性或者GUD，则机床数据
MD 11270:DEFAULT_VALUES_MEM_MASK = 1
必须设置（用于初始化数值的存储器有效）。如果不是这种情况，则发出报警12261“初始化不允许”。

PRLOC	仅有程序局部修改 如果改变一个零件程序、子程序、循环或者Asup中的数据，则在主程序结束之后（用M2,M30等等结束，或者通过面板前方复位键停止）又恢复其原始值。 该属性仅允许用于可编程的设定数据。
	仅在GUD时可以：
SYNR	在读时进刀停止
SYNW	在写入时进刀停止
SYNRW	在读/写时进刀停止
APW	写存取权限
APR	读存取权限 对于机床数据和设定数据，可以后面再改写预设定的存取权。在此允许的数值为 ‘0’ （西门子口令字）至 ‘7’ （钥匙键位置0）。

REDEF的边界条件

只有在定义目标之后才可以修改NC目标的属性。

- 特别是在GUD时必须要注意DEF.../REDEF的顺序。
- （设定数据/系统变量包含在内，并且在处理定义文件之前就已经设定）。
- 首先必须定义该符号（包含在系统之内，或者通过DEF指令定义），接着才可以用REDEF修改。

如果编程了几次属性修改，则最后的修改有效。

数组属性不可以为单个的单元设定，而是始终适用于整个数组：

DEF NCK INT _MYGUD[10,10]	
REDEF _MYGUD INIRE	// ok
REDEF _MYGUD[1,1] INIRE	// 不可以，发出报警 // (数组值)

GUD数组初始化自身不受影响。

```
DEF NCK INT _MYGUD[10] =(0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
DEF NCK INT _MYGUD[100,100] = REP (12)
DEF NCK INT _MYGUD[100,100]  ;
```

带R参数的REDEF指令必须用括号说明。

```
REDEF R[ ] INIRE
```

INI属性

这里必须要注意：在设置变量的INI属性时，必须要有
一个相应大的存储器用于初始值（可以通过
MD18150:MM_GUD_VAL_MEM
设定）。在机床数据
MD 11270:DEFAULT_VALUES_MEM_MASK
中位1必须设置为1
（用于初始化数值的存储器有效）。

太小的存储器会引起报警12261“初始化不允许”。

R参数和系统变量

R参数和系统变量不可以用一个偏离汇编值的缺省值
说明。但是用INIPO、INIRE 或者INICF
可以复位到汇编值。

对于GUD框架数据类型同样（如同已经在定义数据时所
作的说明）不可以用一个偏离汇编值的缺省值说明。



编程举例 1

在GUD时的复位性能

/_N_DEF_DIR/_N_SGUD_DEF	
DEF NCK INT _MYGUD1	；定义
DEF NCK INT _MYGUD2 = 2	
DEF NCK INT _MYGUD3 = 3	

在面板前方按复位键/零件程序结束时初始化：

REDEF _MYGUD2 INIRE	；初始化
M17	

由此在面板前方复位键/零件程序结束"_MYGUD2"时再次设定到值“2”，此时"_MYGUD1" 和 "_MYGUD3"保持它们的值。

编程举例 2

零件程序中的模态转速限制（设定数据）

/_N_DEF_DIR/_N_SGUD_DEF	
REDEF $SA_SPIND_MAX_VELO_LIMS PRLOC	；设定数据，用于界限转速
M17	
/_N_MPF_DIR/_N_MY_MPF	
N10 SETMS (3)	
N20 G96 S100 LIMS=2500	
...	
M30	

在设定数据($SA_SPIND_MAX_VELO_LIMS)极限转速中确定的极限转速为1200转/分钟。因为人们完全可以在一个编排和测试完毕的零件程序中允许一个较高的转速，所以这里编程LIMS=2500。在程序结束之后，通过设定数据设计的值再次生效。

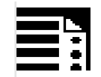
可编程的设定数据 以下的设定数据可以与**REDEF**指令联系起来进行初始化：

序号	名称	GCODE
42000	$SC_THREAD_START_ANGLE	SF
42010	$SC_THREAD_RAMP_DISP	DITS/DITE
43210	$SA_SPIND_MIN_VELO_G25	G25
43220	$SA_SPIND_MAX_VELO_G26	G26
43230	$SA_SPIND_MAX_VELO_LIMS	LIMS
43420	$SA_WORKAREA_LIMIT_PLUS	G26
43430	$SA_WORKAREA_LIMIT_MINUS	G25
43510	$SA_FIXED_STOP_TORQUE	FXST
43520	$SA_FIXED_STOP_WINDOW	FXSW
43700	$SA_OSCILL_REVERSE_POS1	OSP1
43710	$SA_OSCILL_REVERSE_POS2	OSP2
43720	$SA_OSCILL_DWELL_TIME1	OST1
43730	$SA_OSCILL_DWELL_TIME2	OST2
43740	$SA_OSCILL_VELO	FA
43750	$SA_OSCILL_NUM_SPARK_CYCLES	OSNSC
43760	$SA_OSCILL_END_POS	OSE
43770	$SA_OSCILL_CTRL_MASK	OSCTRL
43780	$SA_OSCILL_IS_ACTIVE	OS

可以在零件程序中写的系统变量：

在本说明书的章节15.2中，您可以找到系统变量的列表。所有在零件程序中用W（写）或者WS（写，进刀停止）标记的系统变量可以用RESET指令（复位指令）初始化。

3.10 结构化指令SEFORM，用于步进编辑器(自软件版本 SW 6.4起)



编程

```
SEFORM(STRING[128] 文件名称, INT 级, STRING[128] 图标)
```

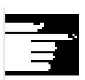
参数说明

SEFORM	用以下参数调用结构化指令： 文件名称，级和图标
文件名称	工作步骤名称
级	主级或者子级索引。 =0　相当于主级 =1, ...　相当于子级1到n
图标	图标名称，显示用于该文件。



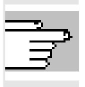
功能

指令SEFORM在步进编辑器中处理，从而从中生成步进画面，用于HMI－高级。步进画面自软件版本SW6.3起可以在HMI－高级中使用，用于更好地读NC子程序。使用结构化指令SEFORM，步进编辑器（以编辑器为基础的程序辅助）通过3个参数支持。



其它说明

- SEFORM指令在步进编辑器中产生。
- 用参数<文件名>变址的字符串与主运行同步，存放在BTSS变量中（与MSG指令类似）。在到下一次SEFROM指令改写之前，该信息保持不变。使用复位键，和零件程序结束时删除该内容。
- 参数级和图标在执行零件程序时由NCK检查，但是不继续处理。



有关编辑器基础上的编程辅助的详细信息，参见：
/BAD/ HMI－高级 操作说明

■

用于记录

保护区



4.1 确定保护区 CPROTDEF, NPROTDEF

编程

```
DEF INT NOT_USED
CPROTDEF(n,t,applim,applus,appminus)
NPROTDEF(n,t,applim,applus,appminus)
EXECUTE \ (NOT_USED)
```

指令说明

DEF INT NOT_USED	局部变量，数据定义为整数类型（参见第10章）
CPROTDEF	定义通道专用保护区 (仅用于 NCU 572/573)
NPROTDEF	定义机床专用保护区
EXECUTE	结束定义

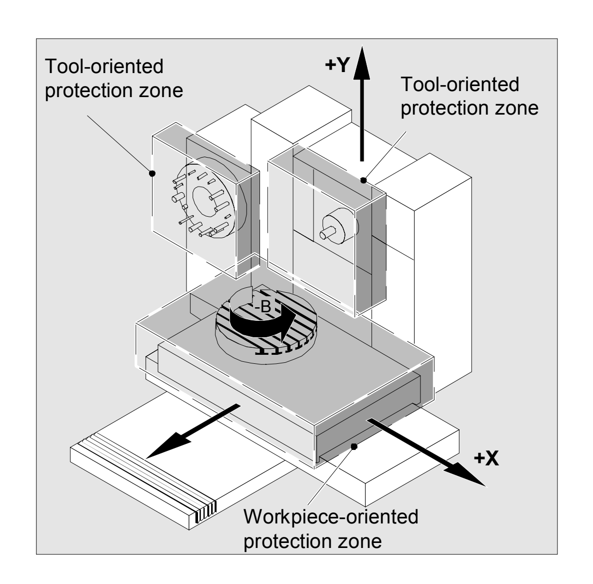
参数说明

n	定义的保护区序号
t	TRUE = 刀具相关的保护区 FALSE = 工件相关的保护区
applim	第3个尺寸的限制方式 0 = 没有限制 1 = 在正方向限制 2 = 在负方向限制 3 = 在正负方向限制
applus	第3个尺寸在正方向限制的值
appminus	第3个尺寸在负方向限制的值
NOT_USED	在有EXECUTE的保护区中故障变量无效



功能

利用保护区，可以保护机床上各个不同的部件、夹具以及工件，防止误动作。

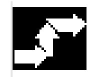


刀具相关的保护区：

隶属于刀具的部件（比如，刀具，刀架）。

工件相关的保护区：

隶属于工件的部分（比如工件，紧固台，夹紧架，主轴卡盘，尾架）。



工作流程

定义保护区

以下部分属于保护区的定义：

- CPROTDEF用于通道专用的保护区
- NPROTDEF用于机床专用的保护区
- 保护区轮廓描述
- 结束EXECUTE定义



在NC零件程序中激活保护区时，可以相对偏移保护区基准点。

轮廓描述基准点

工件相关的保护区在基准坐标系中定义。刀具相关的保护区以刀架基准点F为参考设定。

保护区的轮廓描述

保护区的轮廓在所选择的平面中最多说明11个移动运行。这里，第一个移动运行指轮廓运行。轮廓左侧的区域作为保护区。在CPROTDEF 或者 NPROTDEF 和 EXECUTE 之间出现的位移运行不再执行，而是定义为保护区。

工作平面

在CPROTDEF 或者 NPROTDEF之前用G17、G18、G19选择所要求的平面，并且不允许在EXECUTE之前修改。在 CPROTDEF 或者 NPROTDEF和EXECUTE 之间，不允许编程应用。

轮廓单元

允许：

- G0、G1用于直线轮廓单元
- G2用于顺时针圆弧区段（仅用于工件相关的保护区）
- G3用于逆时针圆弧区段



在SINUMERIK FM-NC中，最多可以使用4个轮廓单元，用于定义一个保护区（最多4个保护区）。
在810D中最多有4个轮廓单元用于定义一个保护区（最多4个通道专用的保护区和4个NCK专用的保护区）。



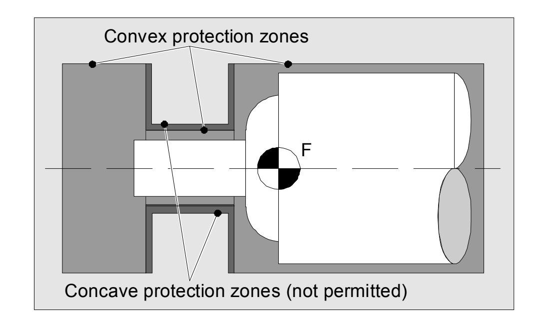
如果要求描述一个整圆作为保护区，则它必须分为两个分圆。不允许使用顺序G2、G3或者G3、G2。有时必须要插进一个较短的G1程序段。

轮廓描述的最后一个点必须与第一个点重叠。

外部保护区（仅可能在工件相关的保护区）必须以顺时针定义。

对于旋转对称的保护区（比如主轴卡盘）必须描述整个轮廓（不仅仅到圆心为止！）。

刀具相关的保护区必须始终为凸面。如果希望有一个凹面保护区，则可以把它分成多个凸面保护区。





在定义保护区时，

- 不允许铣刀半径补偿或者刀沿半径补偿，
- 不允许有转换，
- 不允许框架有效。

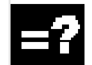
也不允许编程回参考点运行（G74）、固定点返回（G75）、程序段进刀停止或者程序结束。

4.2 激活/取消保护区: CPROT, NPROT



编程

```
CPROT (n,state,xMov,yMov,zMov)
NPROT (n,state,xMov,yMov,zMov)
```



指令和参数说明

CPROT	调用通道专用保护区 (仅用于 NCU 572/573)
NPROT	调用机床专用保护区
n	保护区序号
state	状态参数说明 0 = 取消保护区 1 = 预激活保护区 2 = 激活保护区
xMov,yMov,zMov	偏移几何轴中已经定义的保护区

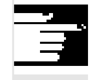


功能

激活事先定义的、监控轮廓冲突的保护区，
取消预激活的或者有效的保护区。

同时在一个通道中有效的保护区的最大数量通过机床
数据确定。

如果没有刀具相关的保护区有效，则按照工件相关的
保护区对刀具轨迹进行检查。



如果没有工件相关的保护区有效，则不进行保护区监
控。



工作流程

激活状态

在通常情况下，在零件程序中用状态＝2激活一个保护区。

状态总是指通道专用的，机床相关的保护区也如此。

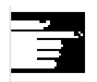
如果通过PLC用户程序规定：可以通过PLC用户程序设置一个保护区有效，则通过状态＝1进行所要求的预激活。

通过状态＝0取消激活，即关闭保护区。不需要偏移。

在（预）激活时偏移保护区

可以用1、2或者3维尺寸偏移。
偏移的参数说明与以下相关：

- 工件专用的保护区中机床零点，
- 刀具专用的保护区中刀架基准点F。



其它说明

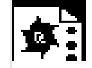
保护区可以在引导及回参考点之后就已经激活。为此必须设置系统变量
`$SN_PA_ACTIV_IMMED [n]` 或者
`$SN_PA_ACTIV_IMMED[n] = TRUE`。
它们必须始终以状态＝2激活，没有偏移。

多次激活保护区

一个保护区也可以同时在几个通道中生效（比如两个相对滑枕中的顶针）。

只有当所有的几何轴都回参考点之后，才可以监控保护区。这里：

- 在一个通道中，保护区不能同时多次激活不同的偏移。
- 机床相关的保护区必须在两个通道中指向相同的方向。

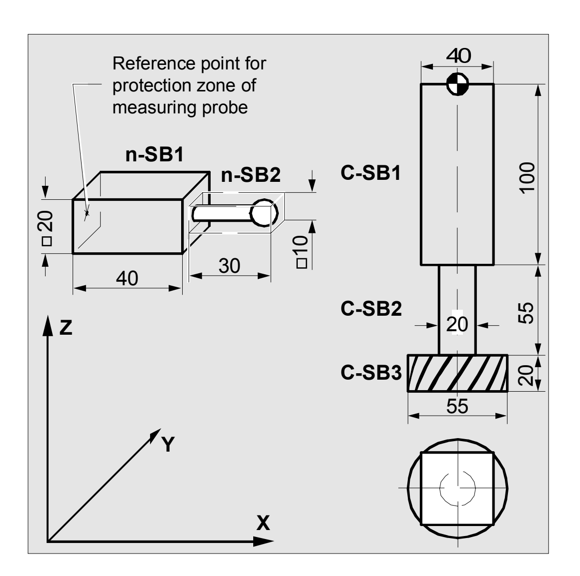


编程举例

铣床中，用测量头监控可能的刀具碰撞。在由于偏移而激活时，应说明测量头的位置。

为此，定义以下的保护区：

- 对于测量头夹持架(n-SB1)和测量头自身(n-SB2)，各有一个机床专用的和刀具专用的保护区。
- 各有一个通道专用的和刀具相关的保护区用于铣刀夹持架(c-SB1)、铣刀柄(c-SB2)和铣刀自身(c-SB3)。



所有保护区定向在Z方向。

在激活时测量头的基准点位置应该在X = –120, Y = 60 和 Z = 80。

`DEF INT SCHUTZB`	定义一个辅助变量
定义保护区	设定方向
`G17`	

`NPROTDEF(1,FALSE,3,10,-10)`
`G01 X0 Y-10`
`X40`
`Y10`
`X0`
`Y-10`
`EXECUTE(SCHUTZB)`
	保护区n–SB1

`NPROTDEF(2,FALSE,3,5,-5)`
`G01 X40 Y-5`
`X70`
`Y5`
`X40`
`Y-5`
`EXECUTE(SCHUTZB)`
	保护区n–SB2

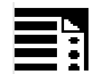
`CPROTDEF(1,TRUE,3,0,-100)`
`G01 X-20 Y-20`
`X20`
`Y20`
`X-20`
`Y-20`
`EXECUTE(SCHUTZB)`
	保护区c–SB1

`CPROTDEF(2,TRUE,3,-100,-150)`
`G01 X0 Y-10`
`G03 X0 Y10 J10`
`X0 Y-10 J-10`
`EXECUTE(SCHUTZB)`
	保护区c–SB2

`CPROTDEF(3,TRUE,3,-150,-170)`
`G01 X0 Y-27,5`
`G03 X0 Y27,5 J27,5`
`X0 Y27,5 J-27,5`
`EXECUTE(SCHUTZB)`
	保护区c–SB3

激活保护区：	
`NPROT(1,2,-120,60,80)`	激活带偏移的保护区n–SB1
`NPROT(2,2,-120,60,80)`	激活带偏移的保护区n–SB2
`CPROT(1,2,0,0,0)`	激活带偏移的保护区c–SB1
`CPROT(2,2,0,0,0)`	激活带偏移的保护区c–SB2
`CPROT(3,2,0,0,0)`	激活带偏移的保护区c–SB3

4.3 检测保护区受损、工作区域限制和软件极限



编程

Status=CALCPOSI(_STARTPOS, _MOVDIST, _DLIMIT, _MAXDIST, _BASE_SYS, _TESTLIM)

说明

状态

0: 功能正常，给定的位移可以完全运行。
–1: 在_DLIMIT中至少有一个分量为负
–2: 在一个转换计算中出现一个错误
如果给定的位移不能完全运行，则送回一个正的、十进制编码的数值：

个位（受损界限种类）：
1: 软件极限限制运行行程。
2: 工作区域极限限制运行行程。
3: 保护区限制运行行程。

如果同时几个界限受到损害（比如软件极限和保护区），则由个位通报界限，由此形成给定运行位移最强烈的限制。

十位

10:
起始值损害界限。
20:
给定的直线直线损害界限。如果终点自身没有损害界限，但是从起始点到终点的行程中会出现极限值受损（比如驶过一个保护区，非线性转换时零件坐标系中弯曲的软件极限），则该值也会被送回。

百位

100:
正的极限值受损（仅当个位为1或者2时，也就是说在软件限位和工作区域限制时）。
100:
一个NCK专用的保护区受损（仅当个位是3时）。
200:
负的极限值受损（仅当个位为1或者2时，也就是说在软件限位和工作区域限制时）。
200: 一个通道专用的保护区受损（仅当个位是3时）。

千位

1000:
系数，用此系数可以乘以该轴的个数，该轴损害极限（仅当个位是1或者2时，也就是说软件极限和工作区域限制时）。
这种轴从1开始计数，在软件极限受损时（个位＝1）与加工轴有关，在工作区域限制受损时（个位＝2）与几何轴有关。
1000: 系数，用此系数乘以受损的保护区的个数（仅当个位是3时）。
如果几个保护区受损，则保护区以百位和千位通报，这将导致给定运行行程的最强烈的限制。

_STARTPOS	起始值，用于横坐标[0]、纵坐标[1]和工件坐标系中应用[2]
_MOVEDIST	增量式位移给定，用于横坐标[0]、纵坐标[1]和工件坐标系中应用[2]
_DLIMIT	[0] - [2]: 分配给几何轴的最小间距。 [3]: 最小间距，在非线性转换时分配此最小间距给一个直线加工轴，如果没有几何轴可以明确地分配时。 [4]: 最小间距，在非线性转换时此最小间距分配给一个旋转加工轴，如果没有几何轴可以明确地分配时。仅在特殊转换时，当软件极限应该受到监控时。
_MAXDIST	数组 [0] - [2] 用于返回值。所有3个几何轴中的增量位移， 加工轴的轴极限没有低于给定的最小间距。 如果运行位移没有限制，则该返回参数的内容等同于_MOVDIST的内容。
_BASE_SYS	FALSE: 或者参数没有说明：在评价位置说明和长度说明时，使用组13(G70, G71, G700, G710; 英制/公制)中的G代码。 在G70有效，并且在公制系统中（或者G71有效，英制系统），提供基准系统中工件坐标系相关的系统变量$AA_IW[X] 和 $AA_MW[X]， 并且有时必须通过功能CALCPOSI进行换算。 TRUE: 在评价位置说明和长度说明时，始终使用控制系统的基准系统，而与组13中有效G代码的数值无关。
_TESTLIM	待检测的界限（二进制编码）： 1: 监控软件极限 2: 监控工作区域限制 3: 监控激活的保护区 4: 监控预激活的保护区 通过数值的相加进行组合。缺省值：15; 检测所有限制。



功能

功能CALCPOSI用于检测几何轴从所给定的起始点起是否可以运行一段给定的位移，而不会损害轴界限（软件极限）、工作区域限制或者保护区。

在给定的位移无法运行时，送回所允许的最大值。
功能CALCPOSI是一个预定义的子程序。
它必须位于一个独立的程序段中。

特殊情况以及其它说明

所有的位移尺寸均是以半径为基准，在G代码“DIAMON”有效时的端面轴时也一样。
如果参见插补的一个轴其位移不能完全运行，则在返回值_MAXDIST中其它轴的位移也要相应减少，从而使所生成的终点位于所给定的轨迹上。

可以允许一个或多个参加插补的轴没有定义软件极限或者工作区域限制或者保护区。
只有当参加插补的轴都回基准后，才监控所有的界限。
如果参加插补的回转轴不是取模轴，则它们才会被监控。

软件极限和工作区域界限监控如同正常运行时一样，取决于有效设定（选择软件极限1或者2的接口信号，
G代码WALIMON / WALIMOF，
设定数据，用于分别激活工作区域界限并确定在监控工作区域界限时是否应该考虑有效刀具的半径）。
在某些运动转换时（比如传送Transmit），加工轴的位置由工件坐标系中的位置不能明确确定（多重意义）。
在正常的运行方式中，在通常情况下只有保证加工轴的连续运行等同于工件坐标系中的连续运行后，其关系才能明晰。因此在这一类情况下，在使用功能CALCPOSI监控软件极限时，当前的加工位置往往会引起多重含义。因此，有时必须在CALCPOSI之前编程一个**STOPRE**，从而可以提供有效的加工轴位置。

在规定的运行行程中运行时，到保护区的间距（在_DLIMIT[3]中规定）不能保证每一处都能遵守。能够保证的仅仅是当延长_MOVDIST中送回的终点时，在这段距离附近不会损害保护区。但是在其曲线过程中，该直线可以任意近地逼近一个保护区。

编程举例

在示例（图）中，在XY平面内软件极限和工作区域均已经标出。
此外定义了三个保护区，两个通道专用的保护区C2和C4以及NCK专用的保护区N3。C2是一个圆弧形状有效的、刀具相关的保护区，半径2毫米。C4是一个正方形的、预激活的和工件相关的保护区，边长为10毫米；N3是一个直角有效的保护区，边长为10毫米或者15毫米。
在下面的NC程序中，首先是定义保护区和工作区域界限，然后以不同的参数调用功能CALCPOSI。CALCPOSI的各个调用结果综合于示例结束处的表格中。


```
N10 def real _STARTPOS[3]
N20 def real _MOVDIST[3]
N30 def real _DLIMIT[5]
N40 def real _MAXDIST[3]
N50 def int _SB
N60 def int _STATUS

N70 cprotdef(2, true, 0) ; 刀具相关的保护区
N80 g17 g1 x-2 y0
N90 g3 i2 x2
N100 i-2 x-2
N110 execute(_SB)

N120 cprotdef(4, false, 0)                ; 工件相关的保护区
N130 g17 g1 x0 y15
N140 x10
N150 y25
N160 x0
N170 y15
N180 execute(_SB)

N190 nprotdef(3, false, 0)                ; 机床相关的保护区
N200 g17 g1 x10 y5
N210 x25
N220 y15
```

```
N230 x10
N240 y5
N250 execute(_SB)
```

```
N260 cprot(2,2,0, 0, 0)                     ; 激活或者预激活保护区
N270 cprot(4,1,0, 0, 0)
N280 nprot(3,2,0, 0, 0)

N290 g25 XX=-10 YY=-10                      ; 定义工作区域界限
N300 g26 xx= 20 yy= 21
N310 _STARTPOS[0] = 0.
N320 _STARTPOS[1] = 0.
N330 _STARTPOS[2] = 0.

N340 _MOVDIST[0] = 35.
N350 _MOVDIST[1] = 20.
N360 _MOVDIST[2] = 0.

N370 _DLIMIT[0] = 0.
N380 _DLIMIT[1] = 0.
N390 _DLIMIT[2] = 0.
N400 _DLIMIT[3] = 0.
N410 _DLIMIT[4] = 0.

; 不同的功能调用
N420 _STATUS = calcposi(_STARTPOS,_MOVDIST, _DLIMIT, _MAXDIST)
N430 _STATUS = calcposi(_STARTPOS,_MOVDIST, _DLIMIT, _MAXDIST,,3)
N440 _STATUS = calcposi(_STARTPOS,_MOVDIST, _DLIMIT, _MAXDIST,,1)

N450 _STARTPOS[0] = 5.                      ; 其它起始点
N460 _STARTPOS[1] = 17.
N470 _STARTPOS[2] = 0.

N480 _MOVDIST[0] = 0.                       ; 其它目标
N490 _MOVDIST[1] =-27.
N500 _MOVDIST[2] = 0.

; 不同的功能调用
N510 _STATUS = calcposi(_STARTPOS,_MOVDIST, _DLIMIT, _MAXDIST,,14)
N520 _STATUS = calcposi(_STARTPOS,_MOVDIST, _DLIMIT, _MAXDIST,, 6)

N530 _DLIMIT[1] = 2.
N540 _STATUS = calcposi(_STARTPOS,_MOVDIST, _DLIMIT, _MAXDIST,, 6)
N550 _STARTPOS[0] = 27.
N560 _STARTPOS[1] = 17.1
N570 _STARTPOS[2] = 0.

N580 _MOVDIST[0] =-27.
N590 _MOVDIST[1] = 0.
N600 _MOVDIST[2] = 0.

N610 _DLIMIT[3] = 2.
N620 _STATUS = calcposi(_STARTPOS,_MOVDIST, _DLIMIT, _MAXDIST,, 12)
```

```
N630 _STARTPOS[0] = 0.
N640 _STARTPOS[1] = 0.
N650 _STARTPOS[2] = 0.
N660 _MOVDIST[0] = 0.
N670 _MOVDIST[1] = 30.
N680 _MOVDIST[2] = 0.
N690 trans x10
N700 arot z45
N710 _STATUS = calcposi(_STARTPOS,_MOVDIST, _DLIMIT, _MAXDIST)
N720 M30
```

示例中测试结果

程序段号 N...	_STATUS	_MAXDIST [0] (= X)	_MAXDIST [1] (= Y)	注释
420	3123	8.040	4.594	保护区SB N3受到损害。
430	1122	20.000	11.429	没有SB－监控，－工作区域界限受到损害。
440	1121	30.000	17.143	仅软件极限监控仍有效。
510	4213	0.000	0.000	起始点损害SB C4
520	0000	0.000	–27.000	预激活的SB C4没有监控。给定的位移可以完全运行。
540	2222	0.000	–25.000	由于_DLIMIT[1]=2，运行位移受到工作区域限制。
620	4223	–13.000	0.000	由于C2和_DLIMIT[3]，到C4的间距共为4毫米。0.1毫米的C2-C3间距不会导致运行位移的限制。
710	1221	0.000	21.213	平移和旋转的框架有效。在_MOVDIST中允许的运行位移适用于偏移的和旋转的坐标系（WCS）中。



其它说明

有关工作区域的详细情况可以参见编程说明PG，软件极限参见功能说明A3。

■

用于记录

特摆动令

5.1 返回编码的位置, CAC, CIC, CDC, CACP, CACN



指令说明

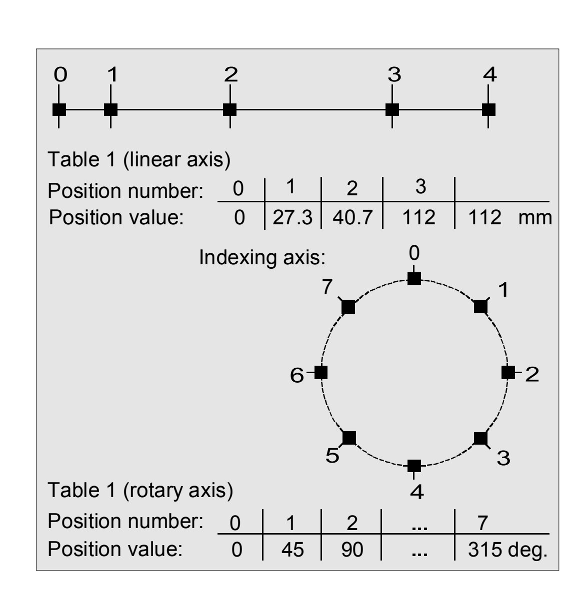
CAC (n)	绝对返回编码的位置
CIC (n)	返回编码的位置，向前n位置（＋）或者向后n位置（－）增量
CDC (n)	以最短的行程返回编码的位置（仅用于回转轴）
CACP (n)	编码的位置绝对在正方向（仅用于回转轴）
CACN (n)	编码的位置绝对在负方向（仅用于回转轴）
(n)	位置序号1,2,…每个轴最多60个位置



过程

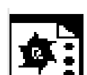
通过机床数据2个轴可以在位置表中分别输入最多60（0到59）个位置。

位置表举例－参见图。



其它说明

如果一个轴位于两个位置之间，则在带CIC(…)的增量数据时不运行。
建议第一个运行指令始终以绝对位置参数编程。



编程举例

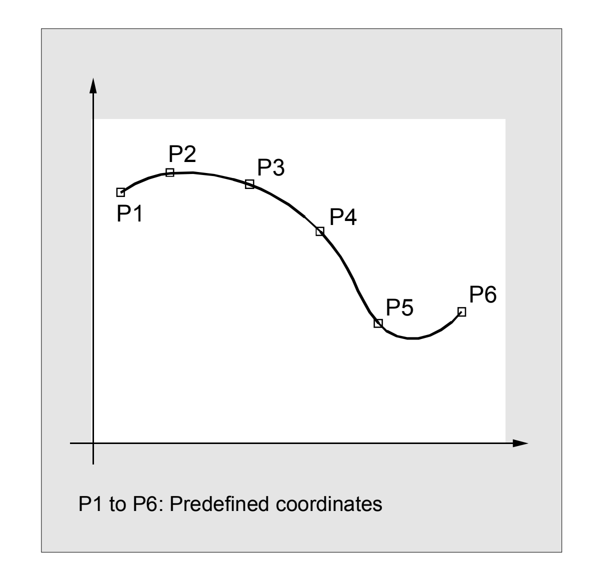
N10 FA[B]= 300	用于位置轴B的进给
N20 POS[B]= CAC (10)	绝对返回编码的位置10
N30 POS[B]= CIC (-4)	从当前位置4返回

5.2 样条插补, ASPLINE, B-/CSPLINE, BAUTO, BNAT, BTAN

序言

通过样条插补可以用平滑的曲线连接各个分散的点。可以使用样条，比如把数字化的点连接成曲线。

不同的样条类型有不同的性能，它们也会产生不同的结果。除了可以选择不同的样条类型外，使用者还可以改变参数的序列。经常需要几次尝试才可以产生所要求的图形。



P1 to P6: Predefined coordinates

编程

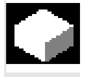
```
ASPLINE X Y Z A B C
```

或者

```
BSPLINE X Y Z A B C
```

或者

```
CSPLINE X Y Z A B C
```



功能

如果要求把一系列点连接成一条曲线，则您可以编程一个样条。

可以有三种类型的样条：

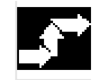
– A样条（Akima样条）
– B样条（非统一、有理的基准样条，NURBS）
– C样条（立体样条）



其它说明

A-,B-和C-样条为模态有效，属于行程指令组。
可以使用刀具半径补偿。通过在平面上的投影进行轮廓冲突监控。

如何对插补轴复合样条由指令SPLINEPATH进行选择（其它信息参见后面几页）。



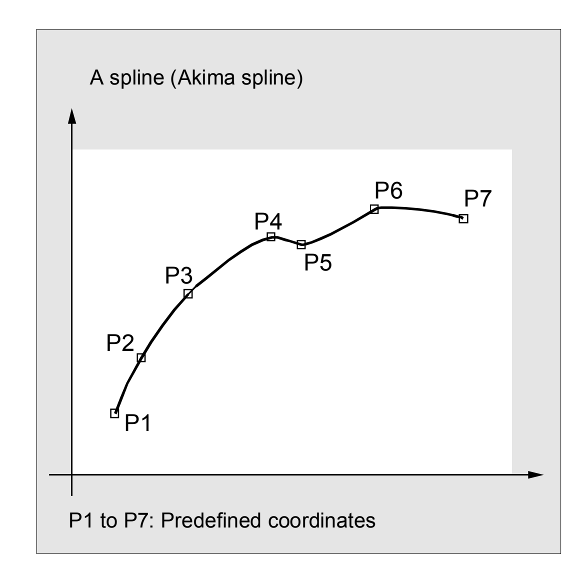
过程

A样条

A样条（Akima样条）精确地通过各个支点。该样条几乎不会产生不好的摆动，但是在各个支点处也不会有曲线过渡。

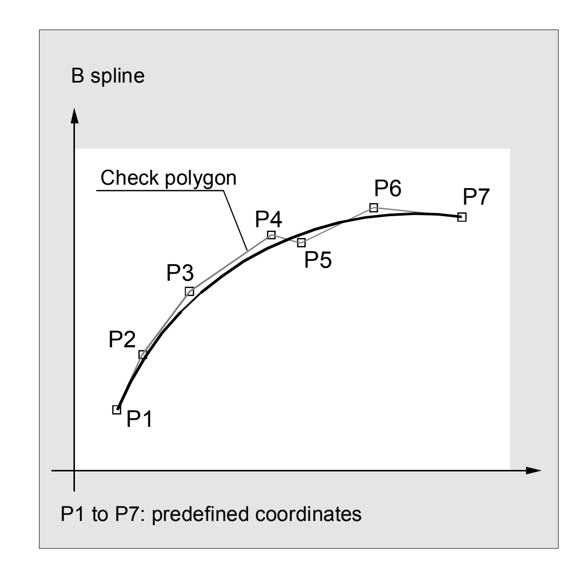
Akima样条为局部的，这就是说如果改变一个支点仅会对附近最多6个相邻的支点产生影响。

因此它主要适用于数字化点的插补。对于该样条可以编程附加条件（详细描述参见后叙）。插补时使用3级多项式。



B样条

在B样条中编程的位置没有支点，而是仅仅有样条的控制点。这就是说样条线不是直接经过这些点，而是通过这些点“拉近”。



通过直线连接这些点，从而构成样条的控制多边形。B样条最适用于描述自由加工面上的刀具位移。它主要是考虑与CAD系统的接口。3级B样条不会产生摆动，尽管它有曲线过渡。

可编程的附加条件对B样条没有影响（其它说明参见后叙）。在起始点和终点处B样条始终与控制多边形轮廓相切。

点加权：

对于每个支点可以有一个加权参数。

编程：

```
PW = n
```

值范围：

0 <= n <= 3; 以0.0001为间隔

作用：

n > 1　样条线由控制点很强地拉近。

n < 1　样条线由控制点较弱地拉近。

样条等级：

正常情况下使用一个3级多项式。也可以使用一个2级多项式。

编程：

```
SD = 2
```

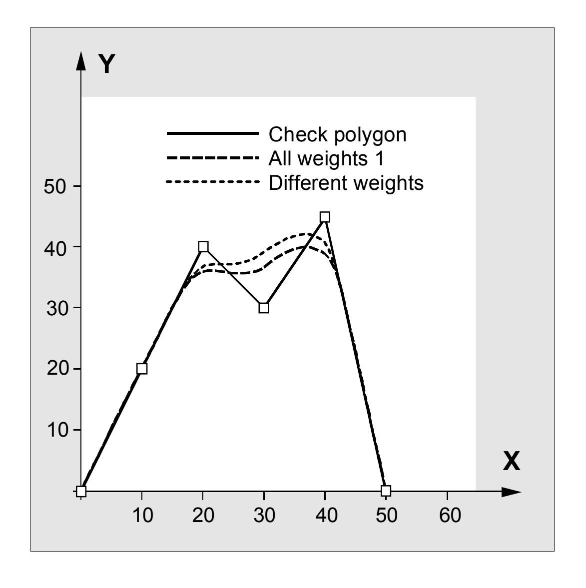


节点间距：

节点间距适合于内部计算。但是控制器也可以处理给定的节点间距。

编程：

PL = 值范围同位移尺寸。



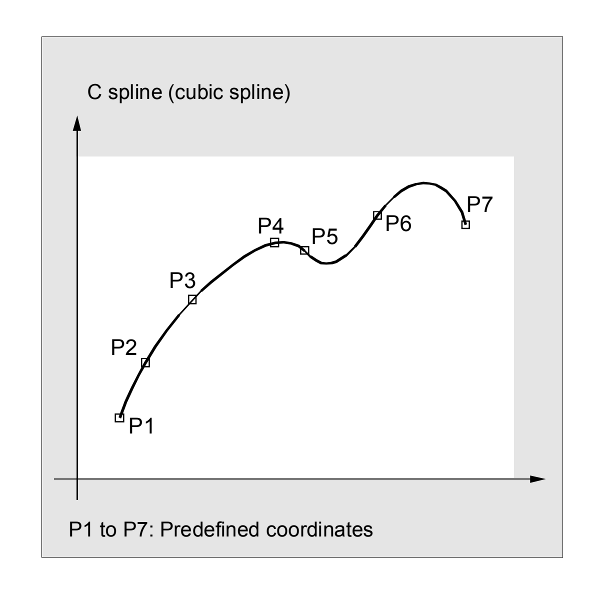
B样条举例：

所有加权1	不同的加权	控制多边形
N10 G1 X0 Y0 F300 G64	N10 G1 X0 Y0 F300 G64	N10 G1 X0 Y0 F300 G64
N20 BSPLINE	N20 BSPLINE	N20 ;entfällt
N30 X10 Y20	N30 X10 Y20 PW=2	N30 X10 Y20
N40 X20 Y40	N40 X20 Y40	N40 X20 Y40
N50 X30 Y30	N50 X30 Y30 PW=0.5	N50 X30 Y30
N60 X40 Y45	N60 X40 Y45	N60 X40 Y45
N70 X50 Y0	N70 X50 Y0	N70 X50 Y0

C样条

与Akima样条不同的是，立体样条（C样条）在支点处有曲线过渡。但是仍有产生摆动的倾向。如果这些点位于一个已知的曲线上，则可以使用该样条。C样条使用3级多项式。

该样条并不是局部的，也就是说改变一个点可能会对许多部分产生影响（强度逐步减弱）。





边界条件：

这些边界条件仅仅适合于Akima样条和立体样条（A样条，C样条）。

通过两组，每组3个指令可以调整样条曲线的过渡性能（起始点或终点）。

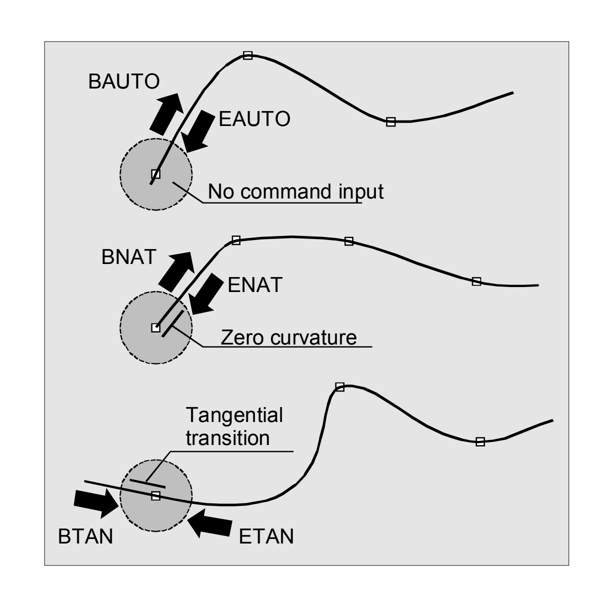
指令说明

开始样条曲线：

BAUTO	没有规定；由第一个点的位置开始
BNAT	曲率零
BTAN	切线过渡到上一段（清除位置）

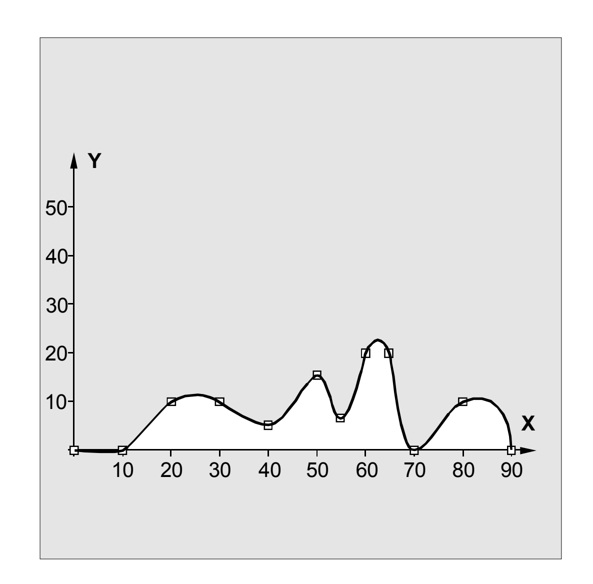
结束样条曲线：

EAUTO	没有规定；从最后一个点的位置结束
ENAT	曲率零
ETAN	切线过渡到下一段（清除位置）



举例

C样条，起始和结束处曲率为零



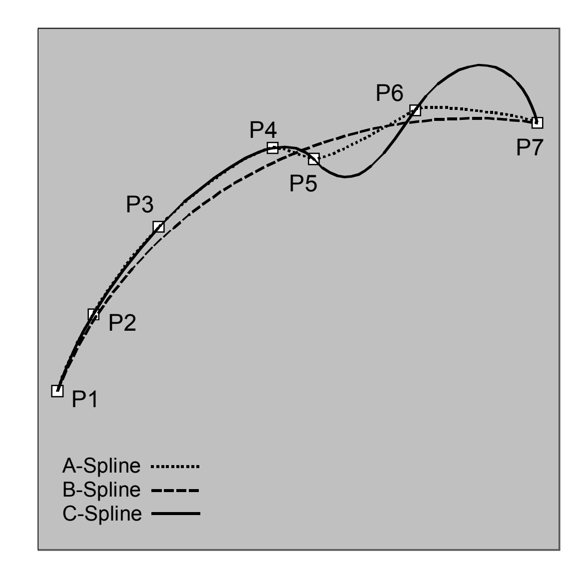
N10 G1 X0 Y0 F300	
N15 X10	
N20 BNAT ENAT	C样条，起始和结束处曲率为零
N30 CSPLINE X20 Y10	
N40 X30	
N50 X40 Y5	
N60 X50 Y15	
N70 X55 Y7	
N80 X60 Y20	
N90 X65 Y20	
N100 X70 Y0	
N110 X80 Y10	
N120 X90 Y0	
N130 M30	



样条的作用？

在相同的支点处对比三种类型的样条：

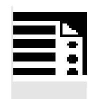
A样条（Akima样条）
样条（Bezier样条）
C样条（立体样条）



样条组合

样条插补中最多可以有8个轨迹轴。使用指令SPLINE PATH可以确定参加样条插补的轨迹轴。这需要在一个独立的程序段中确定。如果SPLINEPATH 没有明确地编程，则通道中开始的3个轴作为样条组合运行。





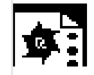
编程

```
SPLINEPATH(n,X,Y,Z,…)
```



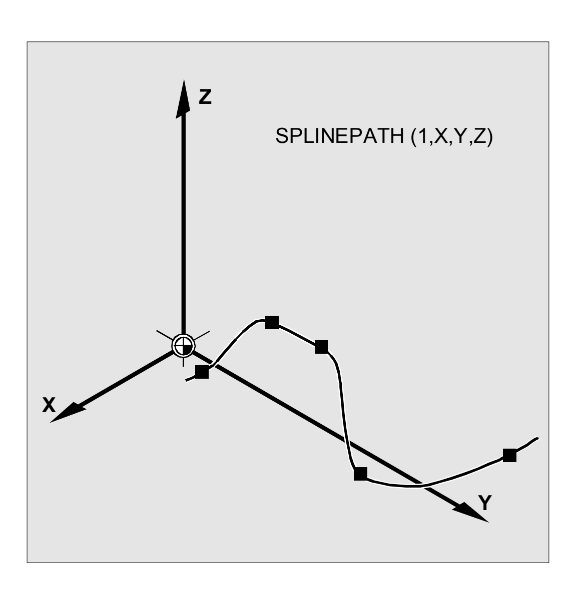
说明

SPLINEPATH(n,X,Y,Z,…)	n = 1, 固定值 X,Y,Z,… 轨迹轴参数说明

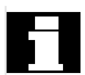


举例

样条组合，有3个轨迹轴



N10 G1 X10 Y20 Z30 A40 B50 F350	
N11 SPLINEPATH(1,X,Y,Z)	样条组合
N13 CSPLINE BAUTO EAUTO X20 Y30 Z40 A50 B60	C样条
N14 X30 Y40 Z50 A60 B70 …	支点
N100 G1 X… Y…	样条插补撤销



样条的调整

使用G指令ASPLINE, BSPLINE 和 CSPLINE，
使样条与程序段终点相联系。

对此必须同时计算一系列的程序段（终点）。

用于计算的缓冲器的大小一般情况下为10个程序段。

并不是每个程序段信息为一个样条终点。控制系统需
要从10个程序段中得到一定数量的样条终点程序段。

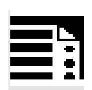
它们是：

A样条：	从10个程序段中必须至少有4个样条程序段。注释程序段和参数计算不在此列。
B样条：	从10个程序段中必须至少有6个样条程序段。注释程序段和参数计算不在此列。
C样条：	在10个程序段中必须至少有机床数据$MC_CUBIC_SPLINE_BLOCKS的内容+1个样条程序段（也就是说正常情况下为9个）。 在机床数据$MC_CUBIC_SPLINE_BLOCKS （缺省值8)中输入点数， 从而计算出样条区段。



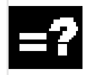
如果超出允许值的范围，则会产生一个报警，同样当
样条插补的轴当作定位轴编程时也会发出一个报警。

5.3 压缩器 COMPOF/ON, COMPCURV, COMPCAD (软件版本SW 6.2)



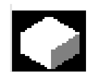
编程

```
COMPON/COMPCURV/COMPCAD
COMPOF
```



说明

COMPON/COMPCURV/COMPCAD	压缩器开
COMPOF	压缩器关



功能

使用G指令COMPON，当插补轴的加速度在程序段过渡时可能产生突变时，速度不变。这可能会导致摆动的生成。

自软件版本 SW4.4起:
使用G指令COMPCURV，可以接通程序段过渡，加速度不变。这样就可以保证程序段过渡时所有轴的速度和加速度的平滑。

自软件版本 SW6.2起:
使用G指令COMPCAD可以选择一个其它的压缩，它可以优化表面质量和速度，这里插补的精度同样可以由机床数据确定。COMPCAD可以压缩计算时间和存储空间。只有当CAD/CAM的程序没有事先采取表面优化的措施时才使用。
特征：

- COMPCAD指令产生加速度恒定的、过渡的多项式程序段。
- 相邻的轨迹中在同一个方向偏离。
- 使用设定数据$SC_CRIT_SPLINE_ANGLE可以确定一个临界角，COMPCAD允许自此临界角起产生棱角。
- 压缩的程序段的个数没有限制为10。
- COMPCAD消除有缺陷的表面过渡。在此继续执行公差范围，但是没有考虑棱角临界角。
- 精磨削功能 G642 照样可以使用。

自软件版本 SW6.3起:

压缩器COMPON, COMPCURV 和

COMPCAD可以扩展，使NC程序也可以遵照所给定的公差压缩。这里NC程序中通过方向矢量编程了方向。

如果有选件方向转换时，才有功能“方向压缩器”。

前面在“使用条件”下所说的限制放宽，即位置值在此也可以通过参数赋值进行。

NC程序段的通常形式：

```
N10 G1 X=<...> Y=<...> Z=<...> A=<...>
       B=<...> F=<...>; 注释
```

轴位置作为参数表达式，带< ...>参数表达式，比如 X=R1*(R2+R3)

在方向转换有效时（TRAORI），在5轴的机床中可以给刀具方向进行如下编程（与运动无关）：

1. 通过以下参数编程方向矢量：
   `A3=< ...> B3=< ... > C3=< ... >`
2. 通过以下参数编程欧拉角或者RPY角：
   `A2=< ...> B2=< ... > C2=< ... >`

只有大圆插补有效时才可以压缩定向运行，也就是说刀具方向的改变在由起始点和终点方向确定的平面中进行。

大圆插补在以下条件下进行：

1. MD 21104:ORI_IPO_WITH_G_CODE = FALSE, 当ORIWKS有效时，并且方向作为矢量编程
   (有 A3, B3, C3 或者 A2, B2, C2)。
2. MD 21104:ORI_IPO_WITH_G_CODE = TRUE, 当 ORIVECT 或者 ORIPLANE 有效时。
   刀具方向可以作为方向矢量编程，或者使用回转轴位置编程。如果一个G代码ORICONxx 或者 ORICURVE有效，或者给方向角(PO[PHI] 和 PO[PSI])编程了多项式，则不进行大圆插补，也就是说这样的程序段没有压缩。

在6轴机床中，除了刀具方向外，还可以编程刀具的旋转。转角用名称THETA (THETA=<...>)编程。
只有当转角线性改变时，编程了旋转的NC程序段才可以进行压缩。也就是说转角不允许编程PO[THT]=(...)多项式。
NC程序段的通常形式：

```
N... X=<...> Y=<...> Z=<...> A3=<...>
      B3=<...> C3=<...> THETA=<...> F=<...>
```

或者

```
N... X=<...> Y=<...> Z=<...> A2=<...>
      B2=<...> C2=<...> THETA=<...> F=<...>
```

如果由回转轴位置说明刀具方向，比如以如下形式：

```
N... X=<...> Y=<...> Z=<...> A=<...>
            B=<...> THETA=<...> F=<...>
```

则以两种不同的方式进行压缩，它由是否进行大圆插补而定。在没有进行大圆插补时，压缩的方向改变通过轴向多项式说明回转轴。

精度

只有允许轮廓偏离编程的轮廓时才可以压缩NC程序段。最大的偏差可以作为压缩器公差在设定数据中设定。允许的公差越大，越多的程序段可以压缩。

轴精度

压缩器给每个轴产生一个样条曲线，它最多与每个轴编程的终点相差一个公差值，该公差值由轴向的机床数据MD设定。

轮廓精度

可以控制轮廓（几何轴）
和刀具方向的最大几何偏差。通过以下的设定参数进
行：

1. 轮廓最大公差。
2. 刀具方向最大角度偏差。
3. 刀具转角最大角度偏差（仅在6轴机床中）。

使用通道专用的机床参数MD20482
COMPRESSOR_MODE可以设定公差值。

0: 轴精度：轴精度轴向公差用于所有轴（几何轴和
方向轴）。

1: 轮廓精度：规定轮廓公差（1.），
方向公差通过轴向公差（a.）。

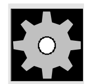
2: 规定最大的角度偏差，用于刀具方向
（2.），轮廓公差通过轴向公差（a.）。

3: 用（1.）规定轮廓公差，用（2.）
规定刀具方向的最大角度偏差。

只有当方向转换(TRAORI)有效时，才可以规定刀具
方向的最大角度偏差。

激活

可以通过指令COMPON, COMPCURV激活
“方向压缩器”（不可以用COMPCAD指令）。
文献： /FB3/, F2:“3－5轴转换”。



机床制造商

有3个机床数据用于压缩器功能：

- $MC_COMPRESS_BLOCK_PATH_LIMIT
  设定一个最大的位移长度，
  直至程序段认为可压缩为止。
  更长的程序段不被压缩。
- $MA_COMPRESS_POS_TOL
  每个轴可以设定一个公差。所形成的样条曲线与编程的终点偏离最多为该设定值。允许的公差越大，越多的程序段可以压缩。
- $MC_COMPRESS_VELO_TOL
  压缩器有效，与FLIN和FCUB相联系，允许的最大轨迹进给偏差可以规定。

在COMPCAD时特别事项：

- $MN_MM_EXT_PROG_BUFFER_SIZE
  应选择很大，比如 100 (kB)。
- $MC_COMPRESS_BLOCK_PATH_LIMIT
  必须明确地较大设定，比如50 (mm) 。
- $MC_MM_NUM_BLOCKS_IN_PREP
  必须设定大于等于60，从而可以明确地加工10个点以上。
- FLIN 和 FCUB 可以不使用。

建议较大的程序段长度，优化的速度：

- $MC_MM_MAX_AXISPOLY_PER_BLOCK = 5
  $MC_MM_PATH_VELO_SEGMENTS = 5
  $MC_MM_ARCLENGTH_SEGMENTS = 10.



CAD/CAM系统通常提供线性程序段，它们执行参数化的精度。
在轮廓比较复杂时这会导致数据量的大幅提高，并可能造成较短的轨迹区段。这种较短的轨迹区段会限制加工速度。
压缩器会使一定数量的（最大10）的这种较短的轨迹区段合并为一个轨迹区段。

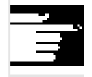
使用模态的G指令COMPON 或COMPCURV 可以执行一个“NC程序段压缩器”。
使用此功能，在线性插补时汇聚了一系列直线程序段（数量限制为10），并且在由机床数据设定的误差公差范围内近似于3级多项式(COMPON) 或者 5级多项式 (COMPCURV)。取代许多较小的程序段，而是加工一个较大的运行程序段。

使用条件：
压缩过程仅在线性程序段（G1）中执行。通过一个其它的NC指令可以中断执行该功能，比如辅助功能的输出，但是参数计算不可中断该功能。
仅包含程序段号、G1、轴地址、进给和注释的程序段才可以进行压缩。所有其它的程序段按原样加工（没有压缩）。变量不可以使用。



例 **COMPON**

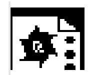
N10 COMPON	或者COMPCURV，压缩器开
N11 G1 X0.37 Y2.9 F600	G1必须位于终点和进给之前
N12 X16.87 Y-4.698 N13 X16.865 Y-4.72 N14 X16.91 Y-4.799 ...	
N1037 COMPOF ...	压缩器关



所有句法简单的程序段被压缩。
比如：

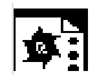
```
N19 X0.103 Y0. Z0.
N20 X0.102 Y-0.018
N21 X0.097 Y-0.036
N22 X0.089 Y-0.052
N23 X0.078 Y-0.067
```

没有压缩的程序段，比如：
扩展地址，如C=100或者A=AC(100)。
自软件版本NC-SW6.3
起具有扩展地址的位移程序段现在也可以压缩。



例COMPCAD

G00 X30 Y6 Z40	
G1 F10000 G642	
SOFT	
COMPCAD	压缩器表面优化开
STOPFIFO	
N24050 Z32.499	
N24051 X41.365 Z32.500	
N24052 X43.115 Z32.497	
N24053 X43.365 Z32.477	
N24054 X43.556 Z32.449	
N24055 X43.818 Z32.387	
N24056 X44.076 Z32.300	
...	
COMPOF	压缩器关
G00 Z50	
M30	



举例“用于方向的压缩器”

在下面的程序示例中，压缩一条用一个多边形轮廓逼近的圆弧。
这里刀具方向与一个圆锥面同步。尽管几个编程的方向改变并不恒定，但是压缩器却生成一个平滑的曲线。

```
DEF INT ANZAHL = 60
DEF REAL RADIUS = 20
DEF INT COUNTER
DEF REAL WINKEL
N10 G1 X0 Y0 F5000 G64
```

最大偏差

```
$SC_COMPRESS_CONTUR_TOL = 0.05
$SC_COMPRESS_ORI_TOL = 5
```

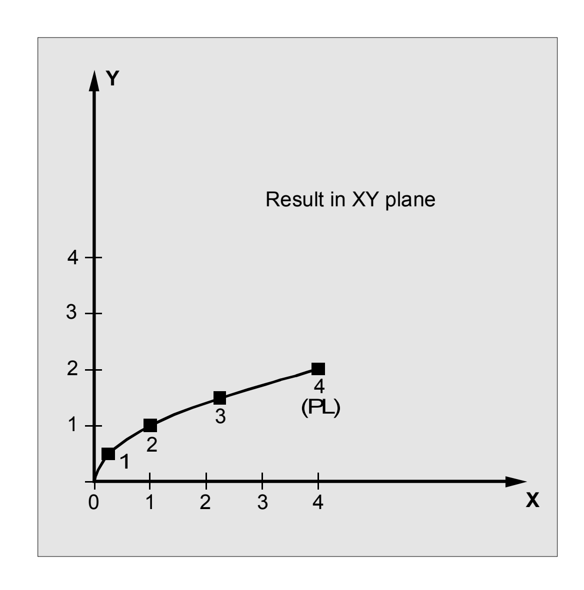
轮廓 0.05 毫米
方向 5 度

```
TRAORI
COMPCURV
N100 X0 Y0 A3=0 B3=-1 C3=1
N110 FOR COUNTER = 0 TO ANZAHL
N120 WINKEL= 360 * COUNTER /ANZAHL
N130  X=RADIUS*COS(WINKEL)Y=RADIUS*
      SIN(WINKEL) A3=SIN(WINKEL)
      B3=-COS(WINKEL) C3=1
N140 ENDFOR
...
```

运行一个由多边形构成的圆弧。
方向以Z轴为中心的圆锥运行。
张角为45度。

5.4 多项式插补 – POLY, POLYPATH (自软件版本 SW 5 起)

控制系统可以按照轨迹运行，这些轨迹中每个所选的轨迹轴有一个功能，在软件版本5之前最多3级的多项式，从软件版本6起则为最大5级多项式。



多项式功能的一般形式为：

$f(p)= a_0 + a_1p + a_2p^2 + a_3p^3$ (至软件版本 SW 5)

或者

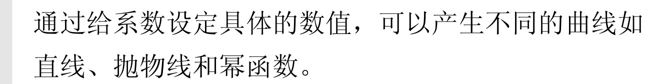
$f(p)= a_0 + a_1p + a_2p^2 + a_3p^3 + a_4p^4 + a_5p^5$

(自软件版本 SW 6起)

这表明：

a_n: 系数常量

p: 参数



通过给系数设定具体的数值，可以产生不同的曲线如直线、抛物线和幂函数。

在设定系数 $a_2 = a_3 = 0$ (至软件版本 SW 5) 或者

$a_2 = a_3 = a_4 = a_5 = 0$ (从软件版本 SW 6起)时

产生比如直线：

$f(p) = a_0 + a_1p$

适用于：

a_0 = 前一程序段结束处的轴位置

a_1 = 在定义范围（PL）

结束处的轴位置和起始位置之间的差值

定义

就其本意来说，多项式插补 (POLY) 与样条插补无关。首先它是用作编程外部产生的样条曲线的接口。在此样条区段可以直接编程。

这种插补方式使NC不用计算多项式系数。当系数直接从CAD系统或者后置处理程序中产生时，可以优化使用。



多项式插补与G0,G1,G2,G3,A样条,B样条和C样条一起在第一个G组中。当它有效时，不需要编程多项式语句：仅使用其名称和终点编程的轴以直线运行到其终点。当所有的轴都如此编程时，控制系统如同G1时一样。

多项式插补通过G组中（比如G0,G1）的一个其它指令关闭。

自软件版本SW5起

子程序调用POLYPATH:

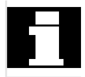
使用POLYPATH可以选择轴组对多项式插补进行说明：

- POLYPATH ("AXES")
  所有的轨迹轴和附加轴。
- POLYPATH ("VECT") 方向轴
  (在方向转换时)。

在正常情况下，编程的多项式也作为多项式用于两个轴组插补。

举例：
POLYPATH ("VECT")
仅选择方向轴用于多项式插补，所有其它的轴以直线运行。

POLPATH ( )
所有轴取消多项式插补。



多项式系数

通过PO值 `(PO[]=)` 或者 `...=PO(...)`
说明一个轴的所有多项式系数。
几个值可以根据多项式等级通过逗号分开。在一个程序段中，可以有不同的多项式等级用于不同的轴。



边界条件：

至软件版本**SW5**

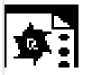
- 几何轴或者轨迹附加轴只有在G0/G1或者POLY有效之后才可以编程多项式，也就是说在圆弧插补时通过多项式运行附加轴是不可能的。
  如果G指令POLY有效，则在正常情况下仅可以编程带PO[…]的多项式。

自软件版本**SW5**起

- 不需要G指令POLY有效，可以编程多项式。在这种情况下，并不是插补编程的多项式，而是每次以线性方式返回到每个轴的编程终点（G1）。
  然后通过编程POLY激活多项式插补。
- 在G指令POLY有效时，可以用预定义的子程序选择POLYPATH (...)，轴应该用多项式插补。

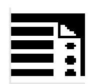
自软件版本**SW6**起

- 系数 a_4 和 a_5 仅从软件版本 SW 6 起才可以。
- 新的、带P0的多项式句法
  到目前为止有效的句法仍然有效。



举例说明带**P0**的有效的多项式句法

到目前为止有效的多项式句法仍然有效。	新的多项式句法（自软件版本SW6起）
PO[轴名称]=(.. , ..)	轴名称=PO(.. , ..)
PO[PHI]=(.. , ..)	PHI=PO(.. , ..)
PO[PSI]=(.. , ..)	PSI=PO(.. , ..)
PO[THT]=(.. , ..)	THT=PO(.. , ..)
PO[]=(.. , ..)	PO(.. , ..)
PO[变量]=IC(.. , ..)	变量=PO IC(.. , ..)



编程

POLY PO[X]=(x_e,a_2,a_3) PO[Y]=(y_e,b_2,b_3) PO[Z]=(z_e,c_2,c_3)

PL=n (至软件版本 SW 5)

POLYPATH("AXES", "VECT")(从软件版本 SW 5起)

扩展到5级多项式和新的多项式句法（自软件版本SW6起）

POLY X=PO(x_e,a_2,a_3,a_4,a_5) Y=PO(y_e,b_2,b_3,b_4,b_5) Z=PO(z_e,c_2,c_3,c_4,c_5) PL=n



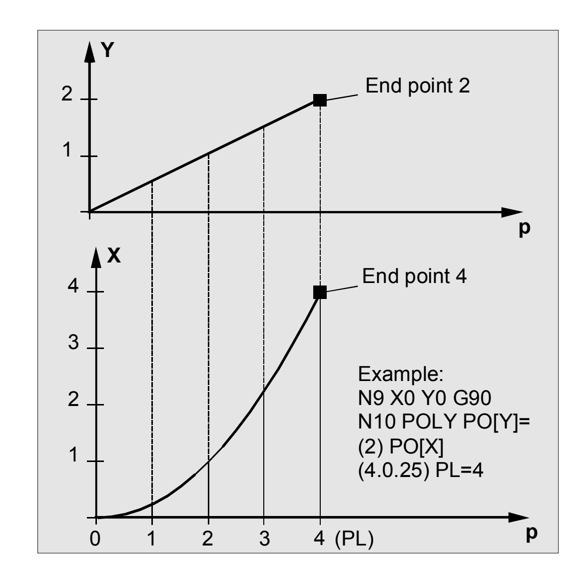
说明

POLY	接通带POLY的程序段的多项式插补。
POLYPATH	多项式插补，可选用于两个轴组AXIS或者VECT
PO[轴名称/变量]=(…,…,…)	终点和多项式系数
X, Y, Z	轴名称
x_e, y_e, z_e	终点位置参数说明，用于各个轴；数值范围同位移尺寸
a_2, a_3, a_4, a_5	系数 a_2, a_3, a_4, 和 a_5 写入值；数值范围同位移尺寸。如果最后的系数值为零，则可以取消。
PL	定义多项式的参数间隔长度（功能f(p)的定义范围）。间隔从0开始，p可以为0到PL之间的数值。PL的理论值范围：0,0001 … 99 999,9999. PL值用于它所在的程序段。如果没有编程PL，则PL=1。

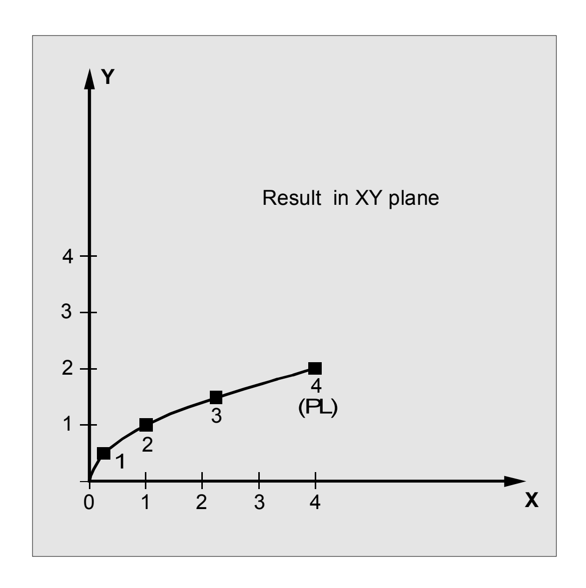
举例

N10 G1 X… Y… Z… F600	
N11 POLY PO[X]=(1,2.5,0.7) -> -> PO[Y]=(0.3,1,3.2) PL=1.5	多项式插补开
N12 PO[X]=(0,2.5,1.7) PO[Y]=(2.3,1.7) PL=3 … N20 M8 H126 …	
N25 X70 PO[Y]=(9.3,1,7.67) PL=5	轴参数说明
N27 PO[X]=(10,2.5) PO[Y]=(2.3)	没有编程PL；在PL=1时起作用
N30 G1 X… Y… Z.	多项式插补关
…	

举例说明**X/Y**平面中的曲线




```
N9 X0 Y0 G90 F100
N10 POLY PO[Y]=(2) PO[X]=(4,0.25) PL=4
```





除数多项式的特殊性

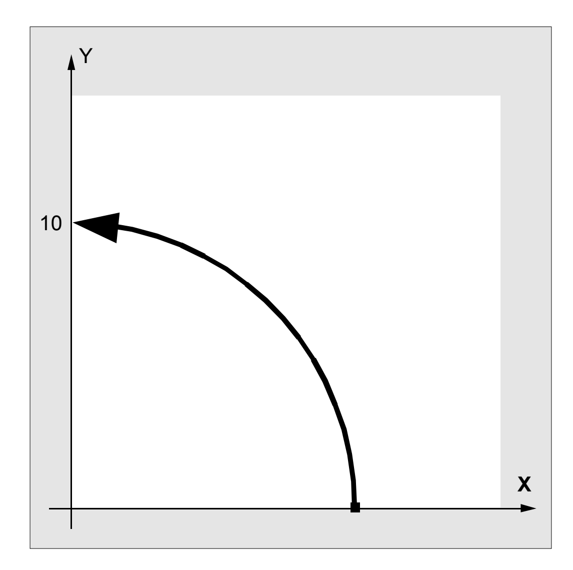
对于几何轴，可以用PO[]=(…)编程一个共同的除数多项式，不用说明轴名称。这就是说几何轴的运行可以作为两个多项式的商进行插补。

由此可以精确描述一个锥体切削（圆弧，椭圆，抛物线，双曲线）。

举例

`POLY G90 X10 Y0 F100`	几何轴直线运行到位置X10,Y0
`PO[X]=(0,-10)  PO[Y]=(10)  PO[]=(2,1)`	几何轴以四分之一个圆弧运行到X0,Y10

除数多项式的系数常数(a_0)始终认为是1，终点与G90/G91无关。

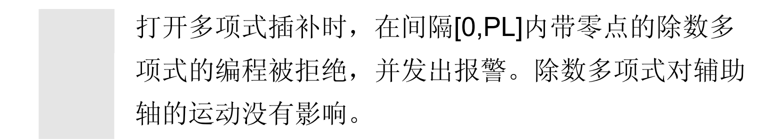


由上面的示例可以得到下面的结果：

$X(p)=10(1–p^2)/(1+p^2)$ 和 $Y(p)=20p/(1+p^2)$
带 $0<=p<=1$

根据编程的起始点、终点、系数a_2 和PL=1
产生如下的中间值：

计数器(X)=$10+0*p–10p^2$
计数器 (Y)=$0+20*p+0*p^2$
除数 = $1+2*p+1*p^2$



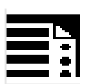
打开多项式插补时，在间隔[0,PL]内带零点的除数多项式的编程被拒绝，并发出报警。除数多项式对辅助轴的运动没有影响。



其它说明

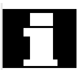
在有G41，G42的多项式插补中可以接通刀具半径补偿，并且可以用于直线或者圆弧插补。

5.5 可设定的轨迹基准, SPATH, UPATH (自软件版本 SW 4.3起)



编程

`SPATH`	FGROUP轴的轨迹基准为弧长。
`UPATH`	FGROUP轴的轨迹基准为曲线参数。



序言

在多项式插补时，用户可以获得两种不同的关系，即决定速度的FGROUP轴 和其它剩余轨迹轴之间的关系。剩余的轨迹轴应该是：

- 或者同步于FGROUP轴的轨迹位移，
- 或者同步于曲线参数 。

到目前为止仅仅实现运动控制的第一个变量，自软件版本4.3 起可以通过一个G指令 (SPATH, UPATH)
选择和编程所要求的性能。



功能

在多项式插补期间 – 在后面描述多项式插补（POLY）时所有的样条插补方式 d (ASPLINE, BSPLINE, CSPLINE) 和线性插补 作为压缩器 (COMPON, COMPCURV) 理解 – 所有轨迹轴的位置 i 通过 $p_i(U)$ 给定。

曲线参数U在一个NC程序段之内从0运行到1，
也经过标准化。

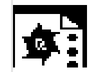
通过语言指令FGROUP可以在轨迹轴之内选择同一个轴，编程的轨迹进给 以该轴为基准。
用速度常数在该轴的位移S中进行插补，表示在多项式插补时进行一个非常数的曲线参数U的修改。

对于FGROUP中不包含的轴有两种方法跟随轨迹：

1. 可以与位移S（SPATH）同步
2. 或者与FGROUP轴（UPATH）的曲线参数U同步

两种轨迹插补方式可以在不同的场合使用，并且可以通过G指令SPATH和UPATH进行切换。

UPATH 和 SPATH 也确定有轨迹运行的F字多项式 (FPOLY, FCUB, FLIN) 的关系。



举例

在下面的示例中，用 G643 精磨削一个20毫米长的正方形。

在此通过机床数据MD33100COMPRESS_POS_TOL[...]
确定每个轴与精确轮廓的最大偏移量。

程序	注释
N10 G1 X… Y… Z… F500	
N20 G643	用G643进行精磨削
N30 XO Y0	
N40 X20 Y0	20毫米棱边长，用于该轴
N50 X20 Y20	
N60 X0 Y20	
N70 X0 Y0	
N100 M30	



边界条件：

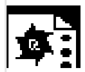
所设定的轨迹基准在下面情况下没有意义：

- 线性- 和圆弧插补,
- 螺纹程序段中
- 当所有的轨迹轴均包含在FGROUP中时



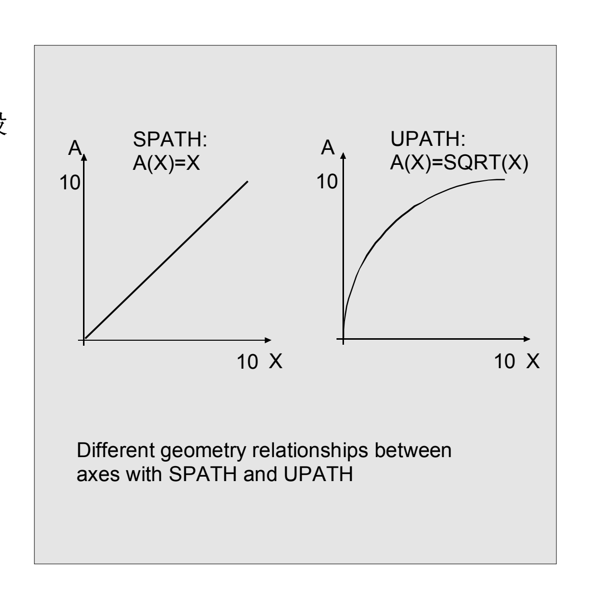
激活

没有包含在FGROUP中的轴，它们的轨迹基准由在第45个G指令组 所包含的两个语言指令SPATH和UPATH 设定。 这些指令模态有效。
当SPATH有效时轴同步于位移，当UPATH有效时则同步于曲线参数。



编程举例

下面的程序示例图示了两种运行控制方式的区别。预设定FGROUP(X,Y,Z)两次有效。



Different geometry relationships between axes with SPATH and UPATH

```
N10 G1 X0 A0 F1000 SPATH
N20 POLY PO[X]=(10, 10) A10
```

或者

```
N10 G1 X0 F1000 UPATH
N20 POLY PO[X]=(10, 10) A10
```

在程序段N20中FGROUP轴的位移S与曲线参数U的平方有关。因此，根据是SPATH有效还是UPATH有效，沿着位移X产生同步轴A的不同的位置：

在上电时控制系统的性能，运行方式更换，复位，程序段搜索，重新定位

在复位之后，通过机床参数MD20150:GCODE_RESET_VALUES [44] 所确定的G代码生效（第45个G代码组）。

精磨削方式的基准值用机床参数MD20150:GCODE_RESET_VALUES [9] 确定 (第10个 G-代码-组)。

机床数据/选件数据

复位之后生效的G代码组由机床参数MD20150:GCODE_RESET_VALUES [44] 确定。
为了保持与当前设备的兼容性，这里预先设定SPATH作为标准值。

精磨削方式的基准值用机床参数MD20150:GCODE_RESET_VALUES [9] 确定 (第10个 G-代码-组)。

轴向机床数据MD33100:COMPRESS_POS_TOL
自软件版本 SW 4.3
起具有扩展的含义：它含有压缩器功能的公差和用G642进行精磨削的公差。

5.6 用开关卡规测量, MEAS, MEAW

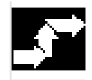
编程

MEAS=±1	G… X… Y… Z…	(+1/+2 测量，带剩余行程删除和上升沿)
MEAS=±2	G… X… Y… Z…	(–1/–2 测量，带剩余行程删除和下降沿)
MEAW=±1	G… X… Y… Z…	(+1/+2 测量，不带剩余行程删除和上升沿)
MEAW=±2	G… X… Y… Z…	(–1/–2 测量，不带剩余行程删除和下降沿)

指令说明

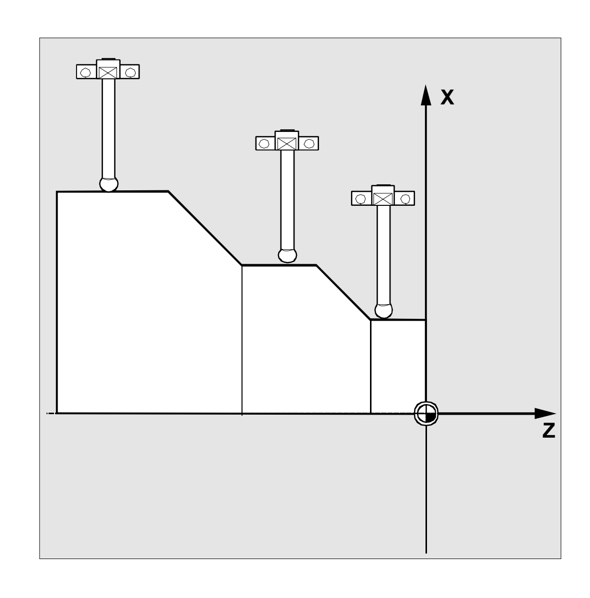
MEAS=±1	在测量输入端1用卡规1测量
MEAS=±2*	在测量输入端2用卡规2测量
MEAW=±1	在测量输入端1用卡规1测量
MEAW=±2*	在测量输入端2用卡规2测量

*根据扩展级最大2个输入端



过程

对于所有在NC程序段中编程的轴，采集测量头开关沿的位置，并且给每个轴写入相应的存储器单元中。最多有2个测量头。



测量结果

每个轴均可以使用下面变量中的测量结果：

- 机床坐标系中在 $AA_MM[轴]中
- 工件坐标系中在 $AA_MW[轴]中

在读这些变量时内部没有进刀停止。
使用 STOPRE 时必须在NC程序中在合适的位置编程一个进刀停止。否则会读入错误的值。

测量任务状态

如果在程序中要求运用判断是否打开测量头，则可以询问 `$AC_MEA[n] (n=` 测量头序号):

0　　没有实现测量任务

1　　测量任务已经顺利完成（测量头已经打开）

如果在程序中偏开测量头，则此变量置为1。在启动一个测量程序段时，该变量自动置为卡规的起始状态。

编程测量程序段, MEAS, MEAW

使用指令MEAS和一种插补方式，返回到工件的实际位置，并接收测量值。在实际位置和给定位置之间的剩余行程删除。

对于一些特殊的测量任务，即必须返回到编程的位置，这种情况下使用功能MEAW。

MEAS和MEAW在程序段中带运行指令编程。至于进给和插补方式(G0, G1, …)
则与当时的测量任务相适应，轴数也是如此。

举例：

```
N10 MEAS=1 G1 F1000 X100 Y730 Z40
```

测量程序段用第一个测量输入端的测量头和直线插补。进刀停止自动产生。

接收测量值

程序段中所有运行的轨迹轴和定位轴的位置被采集
（根据每个控制系统配置的最大轴数）。
在MEAS功能中，测量头开关后运动一定要减速。

注释

如果在一个测量程序段中编程了一个GEO轴，则给所
有当前的GEO轴存储这些测量值。
如果在一个测量程序段中编程了一个进行转换的轴，
则所有进行转换的轴的测量值被存储。



其它说明

功能MEAS和MEAW按程序段方式生效。

5.7 扩展的测量功能 MEASA, MEAWA, MEAC (自软件版本 SW 4起, 选件)

编程

`MEASA\ [轴]=(Modus, TE1,…, TE 4)`	测量，带剩余行程删除
`MEAWA [轴]=(Modus, TE1,…, TE 4)`	测量，不带剩余行程删除
`MEAC [轴]=(Modus, 测量存储器, TE 1,...TE4)`	连续测量，不带剩余行程删除

说明

轴	名称，用于测量所使用的通道轴
模型	运行模态的两位参数说明；由以下构成： **测量模态**（个位）和 0 停止测量任务 1 模态 **1**： 最多有4个不同的同时激活的触发事件 2 模态 **2**： 最多有4个一个接一个的激活的触发事件 3 模态 **3**： 最多有4个一个接一个的激活的触发事件但是在START时没有监控触发事件1（抑制报警21700/21703）。 说明：在MEAC时不可能有模态 3 **测量系统**（十位） 0 或者没有参数说明：有效的测量系统 1 测量系统1 2 测量系统2 3 两个测量系统
TE 1…4	**触发事件** 1 上升沿，测量头1 -1 下降沿，测量头1 2 上升沿，测量头2 -2 下降沿，测量头2
测量存储器	FIFO号（循环存储器）

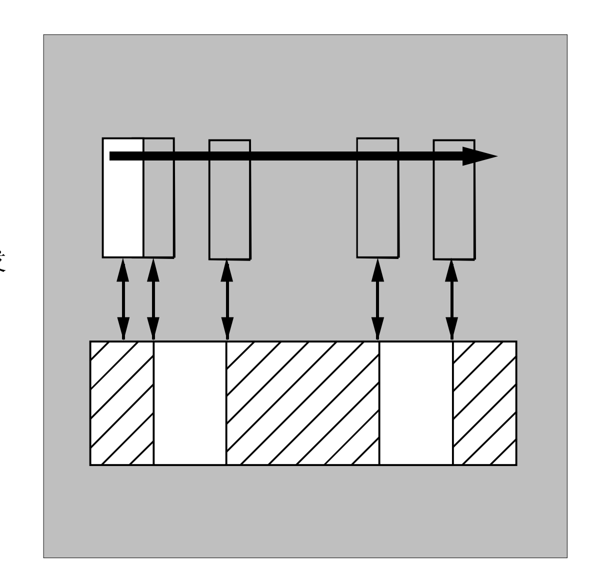


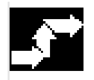
功能

自软件版本SW4起可以使用轴向测量。
在测量时，可以在轴向进行多个测量头和测量系统的测量。

在MEASA、MEAWA时，对于所编程的轴每次测量时最多采集4个测量值，并相应地存储到系统变量的触发事件中。
MEASA和MEAWA以程序段方式生效。

连续的测量任务可以用MEAC进行。在这种情况下测量结果存储到FIFO变量中。同样对于MEAC，每次测量最多有4个测量值。





过程

可以以零件程序编程或者从同步动作（章节10）产生。每个轴在一个时间和同一时间仅可以激活一个测量任务。



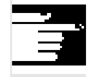
其它说明

- 进给与相应的测量任务相匹配。
- 在**MEASA**和**MEAWA**时，只有在下面的进给时才可以保证结果正确，即进给时每个位置调节周期不会出现超过一个相同的和4个不同的触发事件。
- 在有**MEAC**的连续测量过程中，插补节拍和位置调节节拍之间的比例不可以大于8：1。

触发事件

触发事件由测量头序号和测量信号的触发准则（上升沿或者下降沿）组成。

每次测量时测量头可以处理最多4个触发事件，也就是两个测量头两个测量脉冲沿。
处理的顺序和触发事件的最大个数与所选的模态有关。



同样的触发事件仅可以在测量任务中编程一次（仅适用于模态1）。

工作模态

使用模态的第一个数字选择所希望的测量系统。如果仅有一个测量系统，但是却编程了第二个，则自动使用当前的测量系统。

使用第二个数字，也就是测量模态，匹配测量过程与其控制系统方法。

- **模态 1**：
  触发事件的处理按照其出现的时间顺序进行。
  在这种模态中使用六轴模块时仅可编程一个触发事件，或者在参数说明多个触发事件时自动移植到第二个模态（没有通报）。
- **模态 2**：触发事件的处理按照其编程的顺序进行。
- **模态 3**：
  触发事件的处理按照编程的顺序进行，但是不监控START时的触发事件1。

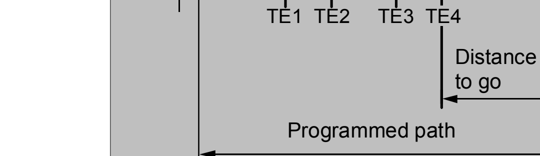


其它说明

在使用2个测量系统时仅可编程两个触发事件。

带和不带剩余行程删除的测量

在编程MEASA时仅在采集了所有要求的测量值之后才删除剩余行程。



对于一些特殊的测量任务，即必须返回到编程的位置，这种情况下使用功能MEAWA。

MEASA和MEAWA可以在一个程序段中编程。

如果MEASA/MEAWA 和 MEAS/MEAW
在一个程序段中编程，则产生一个报警信息。

- MEASA不可以在同步动作中编程。
  但是MEAWA加上剩余行程删除可以作为同步动作编程。
- 如果带MEAWA的测量任务从同步动作中启动，则测量值仅在机床坐标系中可以使用。

测量结果 用于 MEASA, MEAWA

测量结果仅在下面的系统变量中可以使用：

- 在机床坐标系中：

`$AA_MM1[轴]`	在触发事件1时编程的测量系统的测量值
…	...
`$AA_MM4[轴]`	在触发事件4时编程的测量系统的测量值

- 在工件坐标系中：

`$AA_WM1[轴]`	在触发事件1时编程的测量系统的测量值
…	...
`$AA_WM4[轴]`	在触发事件4时编程的测量系统的测量值



其它说明

在读这些变量时内部没有进刀停止。
使用 STOPRE (章节 15.1)
时必须在合适的位置编程一个进刀停止。否则会读入错误的值。

如果启动一个几何轴的轴向测量，则必须给所有剩余的几何轴明确编程同样的测量任务。
这同样适用于进行转换的轴。

举例：

```
N10 MEASA[Z]=(1,1) MEASA[Y]=(1,1)
MEASA[X]=(1,1) G0 Z100;
```

或者

```
N10 MEASA[Z]=(1,1) POS[Z]=100
```



有2个测量系统的测量任务

如果一个测量任务有两个测量系统，则两个可能的触发事件中的每一个均由该轴的两个测量系统采集。规定预留变量的分配：

$AA_MM1[轴]	或者	$AA_MW1[轴]	触发事件1时测量系统1的测量值
$AA_MM2[轴]	或者	$AA_MW2[轴]	触发事件1时测量系统2的测量值
$AA_MM3[轴]	或者	$AA_MW3[轴]	触发事件2时测量系统1的测量值
$AA_MM4[轴]	或者	$AA_MW4[轴]	触发事件2时测量系统2的测量值

测量头状态 通过 $A_PROBE[n]可读
n=测量头
1==测量头偏离
0==测量头没有偏离

MEASA、MEAWA时的测量任务状态

如果程序中要求一个处理，则测量任务状态可以通过
$AC_MEA[n]（n=测量头序号）进行询问。
一个程序段中只要所有编程的触发事件均用测量头“n”
进行测量，则该变量值为1，其它情况下值为0。



如果从同步动作启动测量，则$AC_MEA不再更新。
在这种情况下必须询问新的PLC状态信号DB(31-48)
DBB62 位 3，或者相同值的变量
$AA_MEAACT[”轴”]。

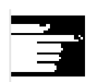
意义：　$AA_MEAACT==1:测量有效
　　　　$AA_MEAACT==0:测量无效

参考文献：　/FB/ M5, 测量

连续测量MEAC

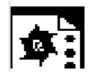
在MEAC时测量值存在于机床坐标系中，并且在所说
明的FIFO[n]存储器中（循环存储器）存储。如果为
了测量设计了两个测量头，则第二个测量头的测量值
存储在一个单独设计的（通过MD可以设定）
FIFO[n+1]存储器中。
FIFO存储器是一个循环存储器，$AC_FIFO变量中循
环原理测量值登记到该循环存储器。

文献：　/PGA/ 章节 10, 同步动作



其它说明

- FIFO内容仅能从循环存储器中读出一次。为了使
  测量数据可以多次重复使用，必须把它们存储在
  用户数据中。
- 如果FIFO存储器中测量值的数量超过机床数据中所
  规定的最大值，则测量自动结束。
- 通过循环读出测量值可以实现无限测量。必须至少以
  新测量值输入的频率进行读出。



编程举例

在模态1中测量，带剩余行程清除

（按事件顺序进行处理）

a) 用 1个测量系统

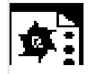
...	
N100 MEASA[X] = (1,1,-1) G01 X100 F100	在模态1中测量，带有效的测量系统在运行行程X=100过程中等待测量头1上升沿/下降沿的测量信号
N110 STOPRE	进刀停止
N120 IF $AC_MEA[1] == FALSE gotof ENDE	控制测量过程
N130 R10 = $AA_MM1[X]	存储属于第一个编程触发事件（上升沿）的测量值。
N140 R11 = $AA_MM2[X]	存储属于第二个编程触发事件（下降沿）的测量值。
N150 ENDE:	



编程举例

b) 带 2个测量系统

...	
N200 MEASA[X] = (31,1-1) G01 X100 F100	在模态1中测量，带两个测量系统在运行行程X=100过程中等待测量头1上升沿/下降沿的测量信号
N210 STOPRE	进刀停止
N220 IF $AC_MEA[1] == FALSE gotof ENDE	控制测量过程
N230 R10 = $AA_MM1[X]	在上升沿时存储测量系统1的测量值。
N240 R11 = $AA_MM2[X]	在上升沿时存储测量系统2的测量值。
N250 R12 = $AA_MM3[X]	在下降沿时存储测量系统1的测量值。
N260 R13 = $AA_MM4[X]	在下降沿时存储测量系统2的测量值。
N270 ENDE:	



在模态2中测量，带剩余行程清除

（按编程顺序进行处理）

程序	注释
...	
N100 MEASA[X] = (2,1,-1,2,-2) G01 X100 F100	在模态2中测量，带有效的测量系统在运行行程X=100过程中等待测量信号，按照如下顺序：测量头1上升沿，测量头1下降沿，测量头2上升沿，测量头2下降沿
N110 STOPRE	进刀停止
N120 IF $AC_MEA[1] == FALSE gotof	控制测量头1的测量过程
MESSTASTER2	
N130 R10 = $AA_MM1[X]	存储属于第一个编程触发事件（测量头1上升沿）的测量值。
N140 R11 = $AA_MM2[X]	存储属于第二个编程触发事件（测量头1上升沿）的测量值。
N150 MESSTASTER2:	
N160 IF $AC_MEA[2] == FALSE gotof ENDE	控制测量头2的测量过程
N170 R12 = $AA_MM3[X]	存储属于第三个编程触发事件（测量头2上升沿）的测量值。
N180 R13 = $AA_MM4[X]	存储属于第四个编程触发事件（测量头2上升沿）的测量值。
N190 ENDE:	



编程举例

模态1中的连续测量：

（按时间顺序进行处理）

测量到100的测量值

...	
N110 DEF REAL MESSWERT[100]	
N120 DEF INT Schleife = 0	
N130 MEAC [X] = (1,1,-1) G01 X1000 F100	在模态1中测量，带有效的测量系统，在$AC_FIFO1下存储测量值，在运行到X=1000的过程中等待测量头1的下降沿测量信号。
N135 STOPRE	
N140 MEAC[X] = (0)	在到达轴位置后停止测量。
N150 R1 = $AC_FIFO1[4]	在参数R1中存储上升测量值的个数。
N160 FOR 循环 = 0 TO R1-1	
N170 测量值[循环] = $AC_FIFO1[0]	读出$AC_FIFO1中测量值，并存储
N180 ENDFOR	

在10个测量值之后测量，带剩余行程删除

...	
N10 WHEN $AC_FIFO1[4]>=10 DO MEAC[x]=(0) DELDTG (x)	剩余行程删除
N20 MEAC[x]=(1,1,1,-1) G01 X100 F500	
N30 MEAC[X]=(0)	
N40 R1=$AC_FIFO1[4]	测量值个数
...	



识别出下面的出错编程，并且显示一个出错：

- MEASA/MEAWA 与 MEAS/MEAW 一起，在一个程序段中编程
  举例：
```
N01 MEAS=1 MEASA[X]=(1,1) G01 F100 POS[X]=100
```
- MEASA/MEAWA 参数个数 <2 或者 >5
  举例:
```
N01 MEAWA[X]=(1) G01 F100 POS[X]=100
```
- MEASA/MEAWA 触发事件不等于 1/ -1/ 2/ -2
  举例:
```
N01 MEASA[B]=(1,1,3) B100
```
- MEASA/MEAWA 为错误模态
  举例:
```
N01 MEAWA[B]=(4,1) B100
```
- MEASA/MEAWA 为两次编程的触发事件
  举例:
```
N01 MEASA[B]=(1,1,-1,2,-1) B100
```
- MEASA/MEAWA 和错误的 GEO轴
  举例:
```
N01 MEASA[X]=(1,1) MEASA[Y]=(1,1) G01 X50 Y50 Z50 F100
```
  GEO轴 X/Y/Z
- 在 GEO轴时不同的测量任务
  举例:
```
N01 MEASA[X]=(1,1) MEASA[Y]=(1,1) MEASA[Z]=(1,1,2) G01
    X50 Y50 Z50 F100
```

5.8 OEM 用户特殊功能, G810 到 G829



OEM地址

OEM用户确定OEM地址的含义。
该功能通过编译循环带来。保留5个OEM地址。
地址名称可以设定。
在每个程序段中允许OEM地址。

OEMIPO1, OEMIPO2

OEM用户可以定义两个附加的G功能名称 OEMIPO1, OEMIPO2 。该功能通过编译循环带来，
保留给OEM用户。

保留的G组

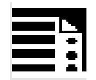
组31，带G810到G819
组32，带G820到G829

可以保留2个G组给OEM用户，每个有10个OEM－G－功能。
这样OEM用户带来的功能可以供外界使用。

功能和子程序

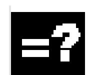
此外，OEM用户也可以通过参数传送设计预定义功能和子程序。

5.9 在拐角处延迟并降低进给率, G62, G621 (自软件版本 SW 6.1起)



编程

```
FENDNORM
G62 G41
G621
```



指令说明

FENDNORM	自动拐角延迟关
G62	在刀具半径补偿有效时，内角处拐角延迟
G621	在刀具半径补偿有效时，所有角处拐角延迟 。



功能

在自动拐角延迟时，在距离拐角很近时以钟形曲线降低进给速度。除此之外，关系到加工的刀具性能的范围可以通过设定数据进行参数设定。它们是：

- 开始和结束进给速度降低
- 进给速度的倍率
- 识别相关角

作为相关角，要考虑其内角小于设定参数所确定的角。

G62仅在内角时有效，通过：

- 有效的刀具半径补偿G41，G42和
- 有效的轨迹控制运行G64，G641

相应的角以降低的进给速度返回,
它由以下公式产生：

F * (用于降低进给速度的倍率) * 进给速度倍率

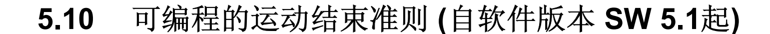
当刀具（以中心点轨迹为基准）在相应角应该变换方向时，表明已经到达了最大可能的进给减速。

在由FGEOUP确定的轴的拐角处，G621作用与G62类似。
使用FENDNORM缺省值，关闭自动拐角倍率的功能。



其它说明

该功能不属于SINUMERIK的标准供货范围，必须有相关的软件版本。
文献： /FBA/ 功能描述。ISO语言



5.10 可编程的运动结束准则 (自软件版本 SW 5.1起)

编程

```
FINEA[<轴>]
COARSEA [<轴>]
IPOENDA [<轴>]
IPOBRKA(<轴>[, [<百分比数值>]])          可以多次说明
ADISPOSA(<轴>, [<Int>][,[<Real>]])      可以多次说明
```

指令说明

FINEA	到达“精准停”后运动结束
COARSEA	到达“粗准停”后运动结束
IPOENDA	到达“插补器停止”后运动结束
IPOBRKA	在制动斜坡中可以更换程序段（自软件版本SW6.2起）
ADISPOSA	运动结束准则的公差窗口大小（自软件版本SW6.4起）
轴	通道轴名称 (X, Y, ...)
百分比数值	在与制动斜坡有关时，程序段转换应以%进行。
Int	方式 0: 公差窗口无效 1: 公差窗口与给定位置相关 2: 公差窗口与实际位置相关
Real	公差窗口大小。与主运行同步，该值登记到设定数据43610:ADISPOSA_ VALUE中。



功能

与轨迹插补（G601,G602和G603）时程序段转换准则类似，在一个零件程序或者指令轴/PLC轴的同步动作中单轴插补时，可以编程运动结束准则。
根据所设定的运动结束准则，零件程序程序段或者带单轴运动的工艺循环程序段可以不同地快速结束。
通过FC15/16/18同样适用于PLC定位指令。

系统变量 $AA_MOTEND

所设定的运动结束准则可以用系统变量
$AA_MOTEND[<轴>]进行询问。

- `$AA_MOTEND[<轴>] = 1` 以“精准停”结束运动。
- `$AA_MOTEND[<轴>] = 2` 以“粗准停”结束运动。
- `$AA_MOTEND[<轴>] = 3` 以“IPO停止”结束运动。
- `$AA_MOTEND[<轴>] = 4` (自软件版本 SW 6.2起) 轴运行制动斜坡程序段更换准则
- `$AA_MOTEND[<轴>] = 5` (自软件版本 SW 6.4起) 制动斜坡中程序段更换，带“给定位置”公差窗口。
- `$AA_MOTEND[<轴>] = 6` (自软件版本 SW 6.4起) 制动斜坡中程序段更换，带“实际位置”公差窗口。



其它说明

在复位之后最后编程的值仍存在。

文献： /FB1/, V1 进给率

自软件版本**SW6.2**起

在制动斜坡中程序段更换准则

与主运行同步，百分比数值登记到
SD 43600:IPOBRAKE_BLOCK_EXCHANGE中。
如果对数值不做说明，则设定数据中的当前值生效。
设定范围是0%到100%。

IPOBRKA中附加的公差窗口

除了已经说明的、制动斜坡中有效的程序段转换准则之外，自软件版本SW6.4起可以选择一个附加的程序段转换准则公差窗口。在下面条件下才可以使能：

- 如果该轴到达目前规定的、制动斜坡的%值，并且
- 自软件版本6.4起，轴当前的实际位置或者给定位置在程序段中不再作为一个公差从轴的终点位置去除。

有关定位轴程序段转换准则的其它信息参见：

文献： /FB2/, P2 定位轴
/PG/, 进给调节和主轴运动



编程示例

```
...
N110 G01 POS[X]=100 FA[X]=1000 ACC[X]=90 IPOENDA[X]
```

以1000转/分钟的轨迹速度和90%的加速度值运行到位置X!00，在到达插补器停止时运动结束

```
...
N120 EVERY $A_IN[1] DO POS[X]=50 FA[X]=2000 ACC[X]=140 IPOENDA[X]
```

当输入端1有效时，以2000转/分钟的轨迹速度和140%的加速度值运行到位置X50，到达插补器停止时运动结束。

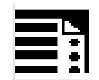
...

制动斜坡程序段转换准则	在零件程序中
	；缺省设定生效
N40 POS[X]=100	
	；当X轴到达位置100和精准停时进行程序段转换
N20 IPOBRKA(X,100)	；激活制动斜坡程序段转换准则
N30 POS[X]=200	；一旦X轴开始制动，进行程序段转换
N40 POS[X]=250	
	；X轴没有在位置200制动，而是继续运行到位置250，；一旦X轴开始制动，进行程序段转换
N50 POS[X]=0	；X轴制动，返回运行到位置0；在位置0和精准停时进行程序段转换
N60 X10F100	
N70 M30	

...

制动斜坡程序段转换准则	在同步动作中
在工艺循环中：	
FINEA	；运动结束准则精准停
POS[X]=100	；当X轴到达位置100和精准停时进行工艺循环程序段转换
IPOBRKA(X,100)	；激活制动斜坡程序段转换准则
POS[X]=100	；POS[X]=100;一旦X轴开始制动，进行工艺循环程序段转换
POS[X]=250	；X轴没有在位置200制动，而是继续运行到位置250，一旦X轴开始制动，进行工艺循环程序段转换
POS[X]=250	；X轴制动，返回运行到位置0；在位置0和精准停时进行程序段转换
M17	

5.11 可编程的伺服参数组 (自软件版本 SW 5.1起)



编程

```
SCPARA [<轴>]= <值>
```



指令说明

SCPARA	确定参数组
轴	通道轴名称(X, Y, ...)
值	所要求的参数组（1<= 值 <=6)



功能

使用SCPARA可以在零件程序和同步动作中编程参数
组（由机床参数MD组成）
（到目前为止仅通过PLC）。

DB3n DBB9 位3

为了避免PLC用户要求与NC用户要求之间产生冲突，
在PLC–>NCK接口上定义另一个位：
DB3n DBB9 位3 “通过SCPARA禁止参数组给定”。



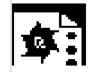
如果仍然对此进行编程，则在SCPARA禁止参数组给
定时不会产生出错报警。

当前的参数组可以通过系统变量 `$AA_SCPAR`[<轴>]
进行询问。



其它说明

- 在软件版本SW5.1之前伺服参数组仅可以通过 PLC (DB3n DBB9 位0–2)进行给定。
  使用G33、G331或者G332时最适合的参数组由控制系统选择。
- 如果在一个零件程序中或者在一个同步动作和 PLC中要求更换伺服参数组，则必须扩展PLC用户程序。
- 文献：　/FB1/V1 进给



编程举例

...	
N110 SCPARA[X]= 3	为x轴选择第三个参数组。
...	

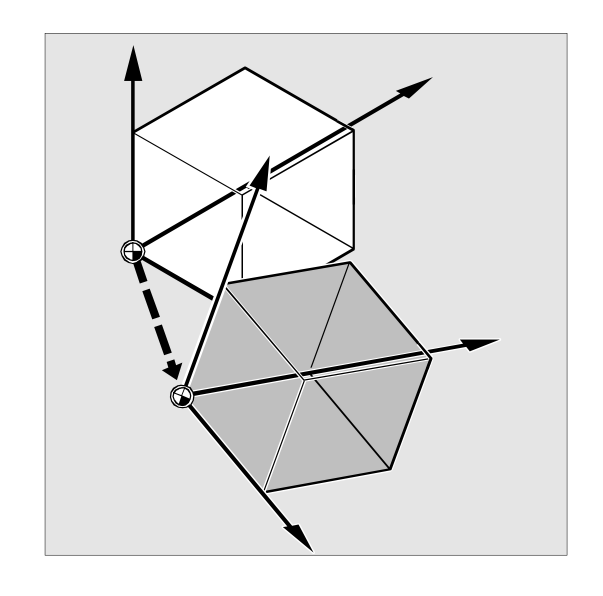
■

框架

6.1 通过框架变量进行坐标转换

用框架变量确定坐标转换

除了在编程说明“基础部分”中所说明的编程方法之外，坐标系也可以用预定义的框架变量确定。



坐标系

所定义的坐标系有以下几种：

MKS: 机床坐标系

BKS: 基准坐标系

BNS: 基准零点坐标系

ENS: 可设定的零点坐标系

WKS: 工件坐标系

预定义的框架变量指什么？

预定义的框架变量已经在控制器的语言中规定了相应的含义，并可以在NC程序中进行处理。

可能的框架变量：

- 基准框架（基准偏移）
- 可设定的框架
- 可编程的框架

框架变量和框架之间的关系

坐标系转换可以通过框架给一个框架变量赋值而激活。

举例： `$P_PFRAME=CTRANS(X,10)`

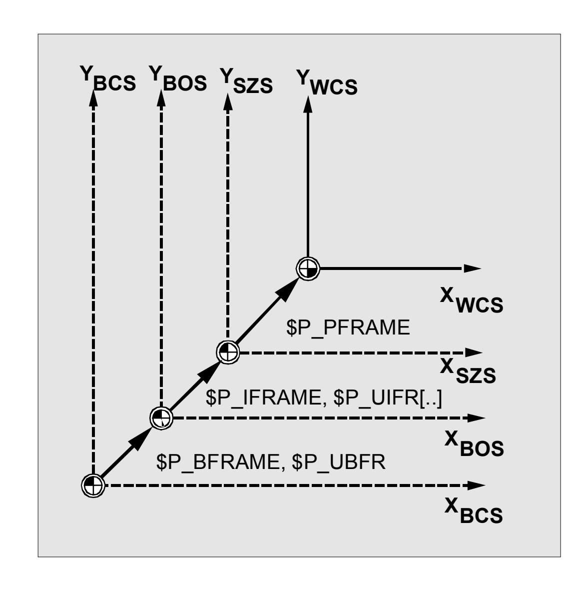
框架变量

`$P_PFRAME` 表明:当前可编程的框架。

框架：

`CTRANS(X,10)` 表明:

X轴可编程的零点偏移为10毫米。



读出实际值

通过零件程序中预定义的变量可以读出坐标系中当前的实际值。

$AA_IM[轴]	在MKS中读出实际值
$AA_IB[轴]	在BKS中读出实际值
$AA_IBN[轴]	在BNS中读出实际值
$AA_IEN[轴]	在ENS中读出实际值
$AA_IW[轴]	在WKS中读出实际值

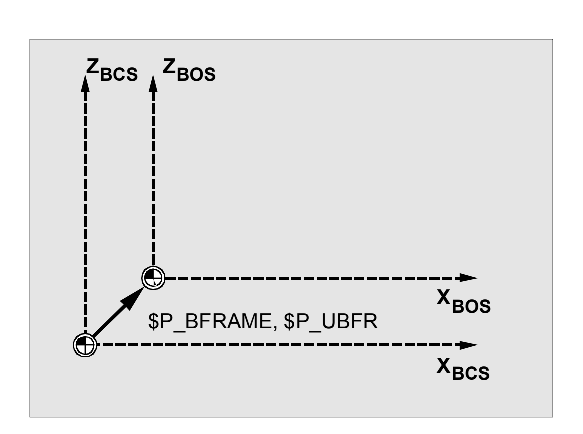
预定义的框架变量 概述

$P_BFRAME

当前的基准框架变量，建立基准坐标系(BKS)和基准零点坐标系(BNS)之间的关系。

如果要使$P_UBFR说明的基准框架在程序中立即生效，则必须做到：

- 编程一个 G500, G54...G599 指令，或者
- $P_BFRAME 用 $P_UBFR 说明。

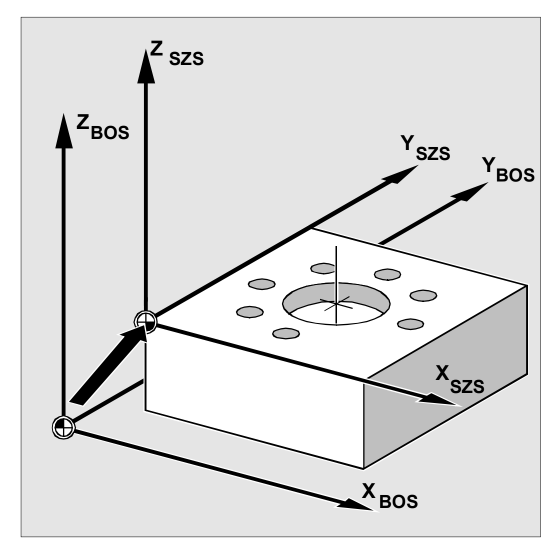


$P_IFRAME

当前可设定的框架变量，建立基准零点坐标系(BNS)和可设定零点坐标系(ENS)之间的关系。

$P_IFRAME 等于 $P_UIFR[$P_IFRNUM]

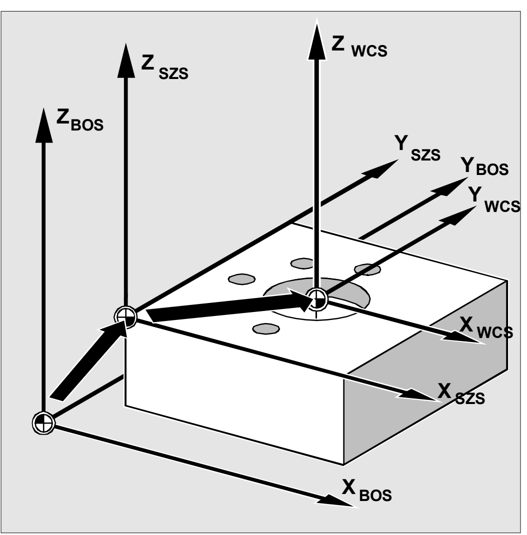
比如在编程G54之后，$P_IFRAME包含由G54定义的平移、旋转、比例和镜像。



$P_PFRAME

当前可编程的框架变量，建立可设定零点坐标系(ENS)和工件坐标系(WKS)之间的关系。

$P_PFRAME 为最终生成的框架,它由编程的TRANS/ATRANS, ROT/AROT, SCALE/ASCALE, MIRROR/AMIRROR 产生，或者由CTRANS, CROT, CMIRROR, CSCALE 对可编程的FRAME赋值形成。



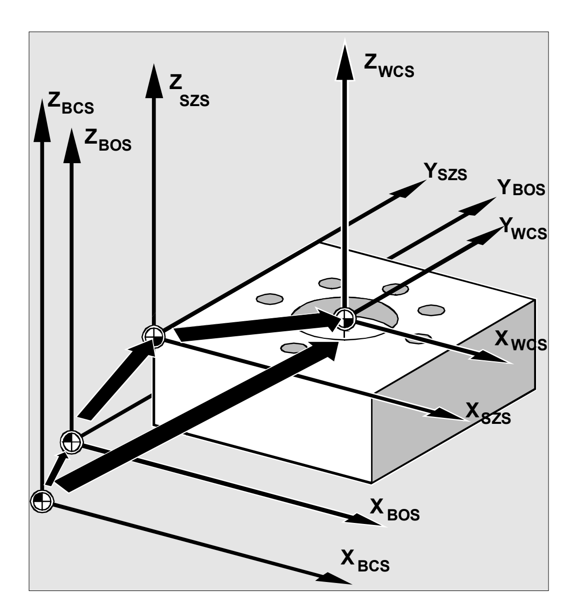
$P_ACTFRAME

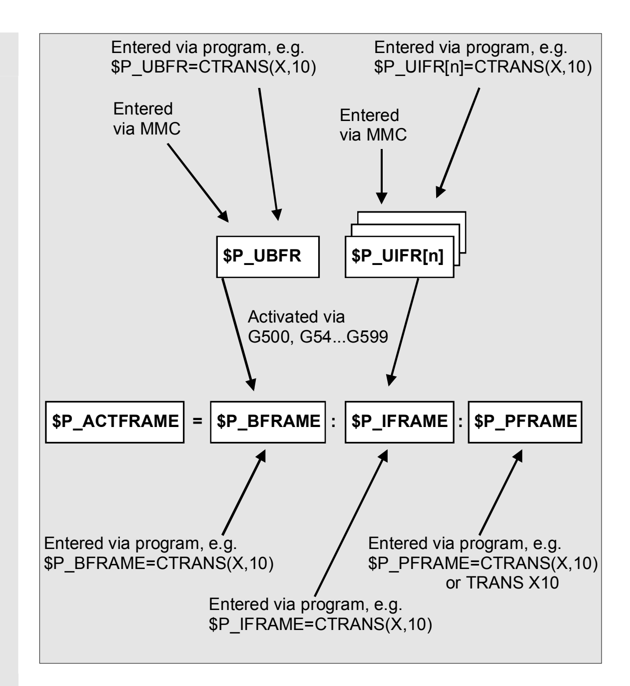
当前所生成的总框架，它由当前的基准框架变量$P_BFRAME，当前可设定的框架变量$P_IFRAME和当前可编程的框架变量$P_PFRAME级联而成。

$P_ACTFRAME说明当前有效的工件零点。

如果 $P_IFRAME, $P_BFRAME 或者 $P_PFRAME 改变, 则 $P_ACTFRAME 重新计算。

$P_ACTFRAME 等于
$P_BFRAME:$P_IFRAME:$P_PFRAME





如果MD20110RESET_MODE_MASK按照如下方式设定，则复位之后基准框架和可设定框架生效：

Bit0=1, Bit14=1 --> $P_UBFR (基准框架) 生效

Bit0=1, Bit5=1 --> $P_UIFR [$P_UIFRNUM]（可设定框架）生效

预定义可设定框架$P_UBFR

通过$P_UBFR编程基准框架，但是不会在零件程序中同时生效。在下面的情况下，用$P_UBFR编写的基准框架一并考虑

- 接通复位，MD RESET_MODE_MASK的位0和14设置
- 执行指令G500, G54...G599。

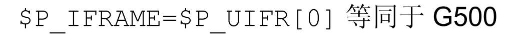
预定义可设定框架$P_UIFR[n]

通过预定义的框架变量`$P_UIFR[n]`，可设定的零点偏移G54到G599由零件程序读或写。

这些变量的结构为FRAME类型数组，单个数字，名称为`$P_UIFR[n]`。

G指令的分配

正常情况下有5个可设定的框架`$P_UIFR[0]`...`$P_UIFR[4]`或者5个相同含义的G指令－G500和G54到G57可以预先设置，在它们的地址中可以存储数值。



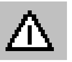
```
$P_IFRAME=$P_UIFR[0] 等同于 G500
$P_IFRAME=$P_UIFR[1] 等同于 G54
$P_IFRAME=$P_UIFR[2] 等同于 G55
$P_IFRAME=$P_UIFR[3] 等同于 G56
$P_IFRAME=$P_UIFR[4] 等同于 G57
```

通过机床数据可以改变框架的个数：

```
$P_IFRAME=$P_UIFR[5] 等同于 G505
… … …
$P_IFRAME=$P_UIFR[99] 等同于 G599
```



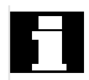
由此可以生成总共100个坐标系，它们可以作为零点用于不同的工装，从而使程序不受影响。



框架变量和框架的编程均要求一个独立的程序段。
例外：用G54,G55,…编程一个可设定的框架时

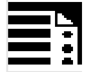


6.2 赋值框架变量和框架



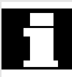
在NC程序中可以直接赋值、框架级联或者框架赋值其它的框架。

直接赋值



编程

```
$P_PFRAME=CTRANS (X, 轴值, Y, 轴值, Z, 轴值, …)
$P_PFRAME=CTOT (X, 角度, Y, 角度, Z, 角度, …)
$P_PFRAME=CSCALE (X, 比例, Y, 比例, Z, 比例, …)
$P_PFRAME=CMIRROR (X, Y, Z)
```

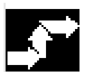


$P_BFRAME 的编程与 $P_PFRAME 的编程类似。



指令说明

CTRANS	在给定轴上的偏移
CROT	围绕给定轴旋转
CSCALE	在给定轴上的比例改变
CMIRROR	在给定轴上的反向



功能

使用这些功能，可以在NC程序中直接给框架或者框架变量赋值。

工作流程

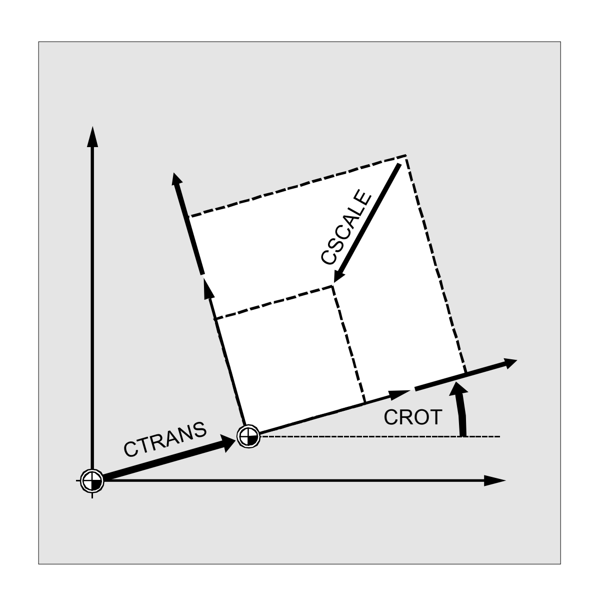
您可以接连编程几个计算。

举例：

```
$P_PFRAME=CTRANS(…):CROT(…):CSCALE…
```

请注意：这些指令必须通过级联运算符冒号(…):(…)相互连接。

由此这些指令首先必须要相互逻辑联系，然后按照编程的顺序加法执行。





其它说明

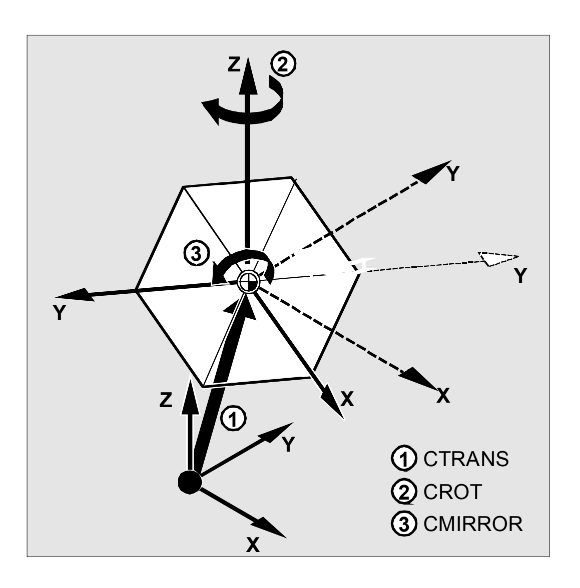
用所给出的指令编程的值赋值给框架并存储。

只有当框架赋值了一个有效的框架变量$P_BFRAME
或者$P_PFRAME后，这些值才有效。



编程举例

通过给当前的、可编程的框架赋值，激活平移、旋转
和镜像。




```
N10 $P_PFRAME=CTRANS(X,10,Y,20,Z,5):CROT(Z,45):CMIRROR(Y)
```

读和修改框架分量



编程（示例）

`R10=$P_UIFR[$P_UIFRNUM, X, RT]`	给变量R10赋值围绕X轴的转角，在当前有效的、可设定的零点偏移$P_UIFRNUM中。
`R12=$P_UIFR[25, Z, TR]`	给变量R12赋值Z轴的偏移值TR，在已经设定的框架号25的数据组中。
`R15=$P_PFRAME[Y, TR]`	给变量R15赋值Y轴的偏移值TR，在当前可编程的框架中。
`$P_PFRAME[X, TR]=25`	在当前可编程的框架中，改变X轴的偏移值TR。X25立即适用。

指令说明

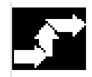
$P_UIFRNUM	使用该变量可以自动建立与当前可设定零点偏移坐标系的联系。
P_UIFR[n, …, …]	通过给出框架号n，从而使用可设定框架n。
 TR FI RT SCMI	对需要读出或者修改的分量的说明： TR平移，FI精确平移，RT旋转，SC比例改变，MI镜像 此外（参见示例）说明相应的轴。



功能

您可以对一个框架的单个的数据进行读写，比如对一个确定的偏移值或者旋转角。

这些值可以修改，或者赋值给另一个变量。



工作流程

调用框架

通过对系统变量$P_UIFRNUM的说明，您可以直接读写用$P_UIFR或者G54, G55, …设定的零点偏移（$P_UIFRNUM包含当前设定框架的序号）。

通过给出相应的序号$P_UIFR[n]，您可以调用所有其它存储的可设定框架$P_UIFR。

对于预定义的框架变量和自身定义的框架，您要给出其名称，比如$P_IFRAME。

数据调用

需读写或者修改的轴名称和框架分量位于方括号中，比如[X, RT] 或者 [Z, MI]。



完整框架的逻辑联系

一个完整的框架可以赋值给另一个框架。



编程（示例）

```
DEF FRAME EINSTELLUNG1
EINSTELLUNG1=CTRANS(X,10)
$P_PFRAME=EINSTELLUNG1
```

自身定义的框架EINSTELLUNG1的数值赋值给当前可编程的框架。

```
DEF FRAME EINSTELLUNG4
EINSTELLUNG4=$P_PFRAME
$P_PFRAME=EINSTELLUNG4
```

当前可编程的框架存储在中间存储器中，在需要时再次返回。



其它说明

RT旋转的数值范围

围绕第1个几何轴旋转： –180° 到 +180°
围绕第2个几何轴旋转： –89.999° 到 +90°
围绕第3个几何轴旋转： –180° 到 +180°

框架级联



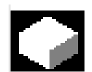
编程（示例）

```
$P_IFRAME=$P_UIFR[15]:$P_UIFR[16]
```

$P_UIFR[15]包含用于零点偏移的数据。接着处理$P_UIFR[16]的数据，比如用于旋转的数据。

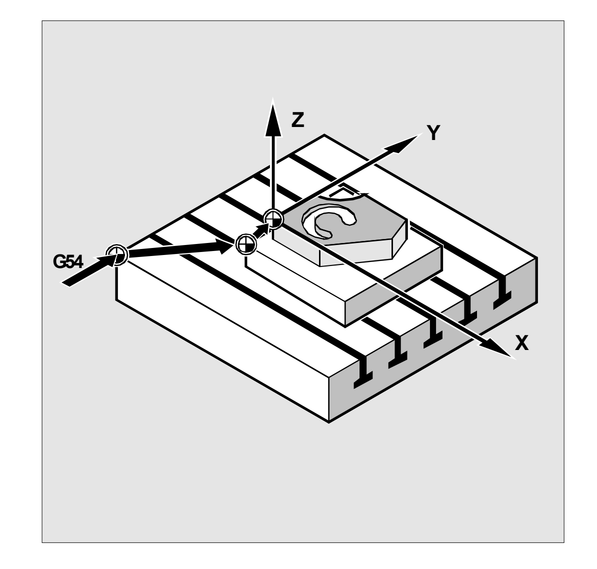
```
$P_UIFR[3]=$P_UIFR[4]:$P_UIFR[5]
```

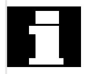
可设定的框架3通过级联可设定的框架4和5产生。



功能

框架级联适用于几个工件的加工，它们装夹在同一个工装夹具中，要求在一个工序中完成加工。





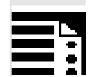
工作流程

框架相互之间以编程的顺序级联，框架分量（偏移，旋转等等）依次相加执行。

说明随行夹具时框架分量可以仅包含某些分值，通过它们的级联产生不同的工件零点。

注意：这些框架必须通过级联操作符冒号：相互联系起来。

定义新的框架



编程

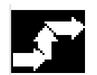
```
DEF FRAME PALETTE 1

PALETTE1=CTRANS(…):CROT(…)…
```



功能

除了前面所说的预定义的、可设定的框架之外，您也可以产生一些新框架。
在此，与FRAME类型变量有关，您可以定义任意名称。



工作流程

使用功能CTRANS, CROT, CSCALE 和CMIRROR，您可以在NC程序中给新的框架赋值。
详细的详细参见前面几页。

确定框架旋转



功能

空间的定向可以根据具体的应用场合通过框架旋转而定。

- ROT:所有几何轴一次旋转
- ROTS, AROTS,
  CROTS:根据空间角度（最大2个）的参数旋转；参见/FB1/ K2中的说明：坐标系。
- TOFRAME:旋转框架"TOFRAME“，其Z轴指向刀具方向。
- TOROT:旋转框架"TOROT"，仅改写已经编程框架的旋转部分。
- PAROT:工件相关的框架旋转旋转部分由可定向刀架的旋转部分确定。



6.3 粗偏移和精偏移



功能

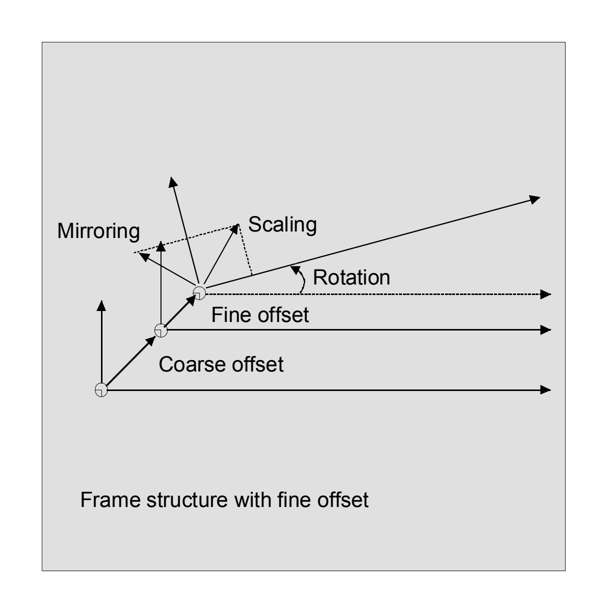
精偏移

使用指令CFINE (X, ..,Y, ...)
可以编程基准框架和所有可设定框架的精偏移。

粗偏移

使用 CTRANS(...) 确定粗偏移。

粗偏移和精偏移相加成为最后的总偏移。

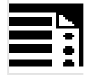


Frame structure with fine offset

编程

```
$P_UBFR=CTRANS(x, 10) : CFINE(x, 0.1) : CROT(x, 45)  ; 各个部分的级联，
                                                     ; 包括偏移、精偏移
                                                     ; 和旋转
$P_UIFR[1]=CFINE(x, 0.5, y, 1.0, z, 0.1)             ; 总框架由
                                                     ; CFINE包括粗偏移
                                                     ; 改写
```

精偏移各个分量的读写由分量参数FI进行。



编程

```
DEF REAL FINEX                    ; 定义变量FINEX
FINEX=$P_UIFR[$P_UIFRNUM, x, FI] ; 读出精偏移，由
                                  ; 变量FINEX进行

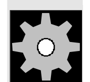
FINEX=$P_UIFR[3, X, FI]           ; 读出X轴的精偏移，
                                  ; 在第3个框架中，通过变量FINEX进行
```

精偏移只有在下面条件下才可以，即MM_FRAME_FINE_TRANS=1时。

只有在激活相应的框架之后，通过操作改变的精偏移才生效，也就是说激活通过G500, G54...G599进行。只要框架生效，激活的框架精偏移就一直有效。



可编程的框架没有精偏移部分。如果带有精偏移的框架赋值给可编程的框架，则总偏移由粗偏移和精偏移的和构成。在读可编程框架时，精偏移始终为零。



机床制造商

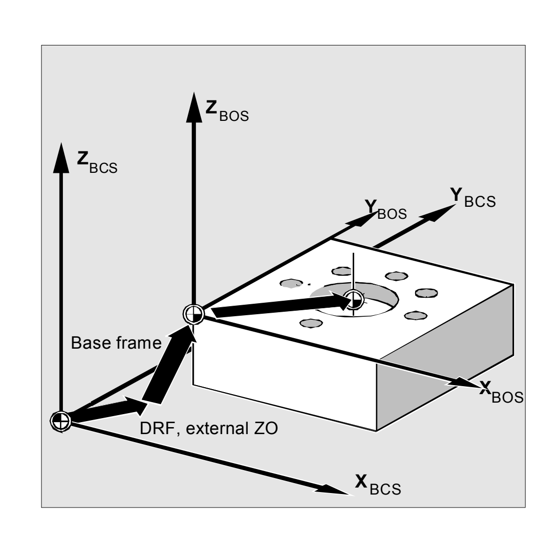
自软件版本**SW5**起
使用MD18600：MM_FRAME_FINE_TRANS可以用以下的变量设计精偏移：
0:
不可以输入精偏移，或者不可以编程精偏移。不可以为G58和G59。
1:
用于可设定框架、基准框架、可编程框架、G58和G59的精偏移可以输入，或者编程。

6.4 DRF-偏移

用手轮偏移, DRF
除了本章中所说的偏移之外，还可以通过手轮（DRF偏移）另外确定一个零点偏移。

DRF偏移在基准坐标系中生效。相互关系见图。

其它信息参见操作说明。

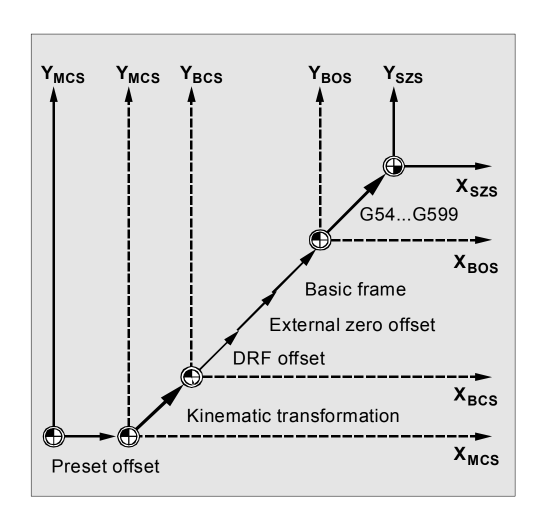




清除DRF偏移，DRFOF
通过DRFOF可以取消该通道中所分配的所有轴的手轮偏移。DRFOF需有一个独立的程序段。



6.5 外部零点偏移



外部零点偏移

这就可以使您在基准坐标系和工件坐标系之间再次进行零点偏移。

在有外部零点偏移时，仅可以编程线性偏移。

编程偏移值，$AA_ETRANS

通过设置轴专用的系统变量进行编程。

偏移值赋值

$AA_ETRANS[Achse]=R_I$

R_I 为REAL类型的计算变量，它包含新的数值。

通常情况下外部偏移不在零件程序中说明，而是由PLC设置。

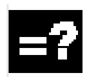
只有当VDI接口（NCU-PLC接口)
设置了相应的信号之后，零件程序中所给的值才会生效。

6.6 编程预设定偏移, PRESETON



编程

```
PRESETON (ACHSE,WERT,…)
```



指令说明

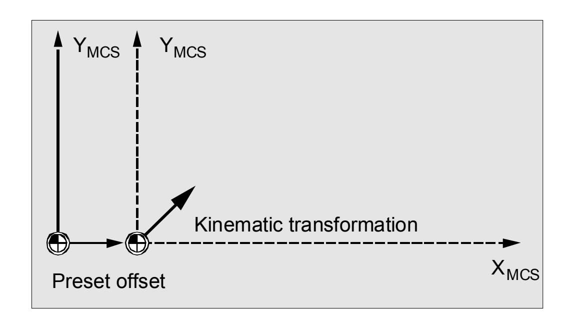
PRESETON	设定实际值
轴	加工轴说明
值	新的实际值，适用于所给定的轴



功能

对于一些特殊的应用场合，有时要求对一个或多个轴的当前位置（停止状态）赋值一个新的、编程的实际值。

说明：用同步动作设定实际值时必须使用字 „WHEN“（如果）或者 „EVEREY“ 。





工作流程

在机床坐标系中赋值实际值，这些值与加工轴有关。

举例：

```
N10 G0 A760
N20 PRESETON(A1,60)
```

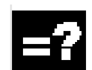
轴A运行到位置760。加工轴A1在位置760处得到新的实际值60。从现在开始在新的实际值系统中定位。



使用PRESETON功能后基准点变得无效。因此这种功能仅仅应用于不需回参考点的轴中。如果需要恢复到原先的系统，则必须使用G74进行回参考点运行 － 参见章节3.1。

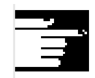


6.7 框架取消,DRFOF, G53, G153 和 SUPA



指令说明

DRFOF	关闭（取消）手轮偏移（DRF）
G53	按照程序段方式取消可编程的和所有可设定的框架
G153	按照程序段方式取消可编程的框架、基准框架和所有可设定的框架
SUPA	按照程序段方式取消所有可编程的框架、基准框架、所有可设定的框架和手轮偏移（DRF）

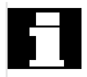


其它说明

可编程的框架按如下方式取消：在可编程框架中赋值
一个“零框架” (没有轴参数) 。
举例：

```
$P_PFRAME=TRANS( )
$P_PFRAME=ROT( )
$P_PFRAME=SCALE( )
$P_PFRAME=MIRROR( )
```

6.8 在空间中通过3个测量点计算框架, MEAFRAME



MEAFRAME是840D语言的一个扩展，用于辅助测量循环。

该功能自软件版本4.3生效。



功能

如果定位一个供加工的工件，则其位置相对于直角的机床坐标系及其理想位置可以偏移或者旋转。

在进行精确加工或者测量时，要么进行精确的物理校准，要么修改零件程序中的运动。

通过在空间探测已知理想位置的三个点可以确定一个框架。使用一个接触式传感器或者光学传感器进行探测，它位于专门的、在支承板上精确固定的孔或者测量球体中。

使用功能MEAFRAME可以从三个理想的点及其相应的测量点计算出框架。

为了使所测量的点通过组合的旋转/平移获得一个理想的坐标，必须使测量点所构成的三角形等同于理想的三角形。这可以通过补偿算法得以实现，也就是说减小偏差的平方和，这里偏差是指测量的值转换到理想的三角形。

实际需要的测量点变形用作衡量测量质量的指示器，因此可以作为MEAFRAME的附加变量输出。



其它说明

由MEAFRAME生成的框架可以通过ADDFRAME功能转换到框架级联中的另一个框架（自软件版本6.3起）。

参见用于其它补偿的MEAFRAME级联应用示例。



编程

```
MEAFRAME IDEAL_POINT,MEAS_POINT,FIT_QUALITY)
```

指令说明

MEAFRAME	在空间中通过3个测量点计算框架
IDEAL_POINT	两维实数数组，它包含理想点的三个坐标
MEAS_POINT	两维实数数组，它包含理想点的三个坐标
FIT_QUALITY	实数变量，下面的信息可以由此返回： -1: 这些理想点位于直线附近：该框架不可以计算。送回的框架变量包含一个中性框架。 -2: 这些理想点位于直线附近：该框架不可以计算。送回的框架变量包含一个中性框架。 -4: 旋转矩阵的计算由于其它其它原因失败 正值： 变形之和（点之间的距离）， 需要此变形之和，用于把所测量的三角形转换到 一个理想的三角形。

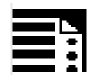


应用示例

```
; 零件程序1
;
DEF FRAME CORR_FRAME
;
; 设置测量点
DEF REAL IDEAL_POINT[3,3] = SET(10.0,0.0,0.0,   0.0,10.0,0.0,  0.0,0.0,10.0)
DEF REAL MEAS_POINT[3,3]  = SET(10.1,0.2,-0.2, -0.2,10.2,0.1, -0.2,0.2, 9.8);         用于测试
DEF REAL FIT_QUALITY = 0
;
DEF REAL ROT_FRAME_LIMIT   = 5;         允许零件位置最大5°旋转
DEF REAL FIT_QUALITY_LIMIT = 3;         在理想三角形和测量三角形之间允许最大3mm的偏移
DEF REAL SHOW_MCS_POS1[3]
DEF REAL SHOW_MCS_POS2[3]
DEF REAL SHOW_MCS_POS3[3]
; ========================================================
;
N100 G01 G90 F5000
N110 X0 Y0 Z0
;
```

```
N200 CORR_FRAME=MEAFRAME(IDEAL_POINT,MEAS_POINT,FIT_QUALITY)
;
N230 IF FIT_QUALITY < 0
SETAL(65000)
GOTOF NO_FRAME
ENDIF
,
N240 IF FIT_QUALITY > FIT_QUALITY_LIMIT
SETAL(65010)
GOTOF NO_FRAME
ENDIF
;
N250 IF CORR_FRAME[X,RT] > ROT_FRAME_LIMIT;          限制第一个RPY角
SETAL(65020)
GOTOF NO_FRAME
ENDIF
;
N260 IF CORR_FRAME[Y,RT] > ROT_FRAME_LIMIT;          限制第二个RPY角
SETAL(65021)
GOTOF NO_FRAME
ENDIF
;
N270 IF CORR_FRAME[Z,RT] > ROT_FRAME_LIMIT;          限制第三个RPY角
SETAL(65022)
GOTOF NO_FRAME
ENDIF
;
N300 $P_IFRAME=CORR_FRAME;          用一个可设置的框架激活探测框架
;
; 通过定位几何轴检测框架的理想点
;
N400 X=IDEAL_POINT[0,0] Y=IDEAL_POINT[0,1] Z=IDEAL_POINT[0,2]
N410 SHOW_MCS_POS1[0]=$AA_IM[X]
N420 SHOW_MCS_POS1[1]=$AA_IM[Y]
N430 SHOW_MCS_POS1[2]=$AA_IM[Z]
;
N500 X=IDEAL_POINT[1,0] Y=IDEAL_POINT[1,1] Z=IDEAL_POINT[1,2]
N510 SHOW_MCS_POS2[0]=$AA_IM[X]
N520 SHOW_MCS_POS2[1]=$AA_IM[Y]
N530 SHOW_MCS_POS2[2]=$AA_IM[Z]
;
N600 X=IDEAL_POINT[2,0] Y=IDEAL_POINT[2,1] Z=IDEAL_POINT[2,2]
N610 SHOW_MCS_POS3[0]=$AA_IM[X]
N620 SHOW_MCS_POS3[1]=$AA_IM[Y]
N630 SHOW_MCS_POS3[2]=$AA_IM[Z]
;
N700 G500;        取消可设定框架，因为已经用零框架（没有填入数值）预置
;
NO_FRAME:
M0
M30
```





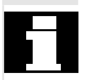
框架级联的应用示例

级联MEAFRAME用于补偿

功能MEAFRAME( )提供一个补偿框架。
如果补偿框架与可设定框架$P_UIFR[1]级联，它在功能调用时有效，比如G54，则可以获得一个可设定框架，用于运行或加工时的其它换算。

用ADDFRAME 级联（自软件版本6.3起）

如果在框架级联中要求该补偿框架在一个其它的地方生效，或者在可设定框架之前还有其它的框架有效，则可以用功能ADDFRAME( )级联到一个通道基准框架或一个系统框架。



在这些框架中，以下功能不可生效：

- 用MIRROR镜像
- 用SCALE改变比例

用于给定值和实际值的输入参数为工件坐标。在控制器的基准系统中，这些坐标始终

- 为公制或者英制（G71/G70），并且作为
- 与半径相关的（DIAMOF）

尺寸说明。



其它说明

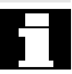
有关FRAME级联的其它信息参见/FB/K2，轴，坐标系，框架

6.9 NCU 全局框架 (自软件版本 SW 5起)



功能

对于所有的通道，每个NCU仅有一个NCU全局框架。
NCU全局框架可以由所有的通道读写。分别在各个通道中激活NCU全局框架。
通过全局框架，通道轴和加工轴可以偏移、改变比例和镜像。
在全局框架中各个轴之间没有几何关系。因此不可以进行旋转和编程几何轴名称。



- 全局框架中不可以使用旋转。编程旋转时会产生报警："18310 通道 %1 程序段 %2 框架:不可以旋转”
- 可以进行全局框架和通道专用框架的级联。
  最后生成的框架包含所有的框架分量，包括用于所有轴的旋转。如果带旋转分量的框架赋值于一个全局框架，则产生报警“框架：不可以旋转”。

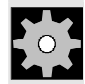
NCU 全局基准框架: $P_NCBFR[n]
最多可以设计8个NCU全局基准框架。

机床制造商

全局的基准框架的个数通过MD设定。
（参见/FB/K2，轴，坐标，框架）通道专用的基准框架可以同时存在。

全局框架可以由一个NCU的所有通道读写。在写全局框架时，由用户考虑通道的协调。比如，这可以通过等待标记（WAITMC）实现。

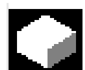
NCU 全局可设定框架：$P_UIFR[n]
所有可设定框架 G500, G54...G599
既可以设计为NCU全局框架，也可以设计为通道专用框架。



机床制造商

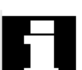
所有可设定框架可以通过 MD 18601
MM_NUM_GLOBAL_USER_FRAMES
改变为全局框架。
参见/FB/K2，轴，坐标，框架。
使用框架的编程指令时，可以使用通道轴名和加工轴名作为轴名称。编程几何轴名称时会出现报警，从而无法进行。

6.9.1 通道专用框架



功能

通过MD28081 MM_NUM_BASE_FRAMES
可以设定通道中基准框架的个数。在进行标准配置时，要保证每个通道至少有一个基准框架。每个通道最多可以有8个基准框架。在通道中除了8个基准通道之外，还可以有另外8个NCU全局基准框架。



可设定框架或者基准框架可以

- 通过零件程序和
- 通过机床控制面板

由操作和PLC读写。

精偏移也可以用于全局框架。

全局框架也可以抑制，如同通道专用框架通过G53, G153, SUPA 和 G500进行一样。

$P_CHBFR[n]

通过系统变量$P_CHBFR[n]可以读写基准框架。在写一个基准框架时，没有激活级联的总基准框架，而是执行一个G500,G54..G599指令后才激活。变量主要作为基准框架中用于写过程的存储器，由MMC或者PLC进行。这些框架变量通过数据存储进行保护。

通道中的第一个基准框架

写入预定义的变量 $P_UBFR

时并不立即激活带数组0的基准框架，而是在执行 G500, G54..G599中的一个指令后才激活。

变量也可以在程序中读写。

$P_UBFR

$P_UBFR 与 $P_CHBFR[0]一致。

正常情况下在通道中始终会有一个基准框架，从而保证系统变量与旧版本兼容。如果没有通道专用基准框架，则在读写时会产生报警“指令不允许”。

6.9.2 在通道中起作用的框架



功能

自软件版本**SW6.1**起
当前的系统框架 用于
$P_PARTFRAME TCARR 和 **PAROT**
$P_SETFRAME 设置实际值和刮痕,
$P_EXTFRAME 外部零点偏移,
通过这些系统变量可以在零件程序中读写当前的系统框架。

$P_NCBFRAME[n]
当前的**NCU**全局基准框架
通过系统变量 $P_NCBFRAME[n]
可以读写当前的全局基准变量数组单元。在通道中写过程中，最后生成的总基准框架一起计算在内。
修改的框架仅在编程的通道中生效。如果要求修改一个NCU所有通道的框架，则必须同时说明
$P_NCBFR[n] 和 $P_NCBFRAME[n]。
然后，其它的通道还必须激活比如带G54的框架。在写一个基准框架时，重新计算总的基准框架。

$P_CHBFRAME[n]
当前通道的基准框架
通过系统变量 $P_CHBFRAME[n]
可以读写当前通道基准框架的数组单元。在通道中写过程中，最后生成的总基准框架一起计算在内。在写一个基准框架时，重新计算总的基准框架。

$P_BFRAME

通道中当前的第一个基准框架

通过预定义框架变量 $P_BFRAME

可以读写零件程序中当前的基准框架，它是通道中有效的、带数组0的基准框架。写入的基准框架立即计算在内。

$P_BFRAME 与 $P_CHBFRAME[0]一致。

在正常情况下，系统变量始终有一个有效值。如果没有通道专用基准框架，则在读写时会产生报警“框架：“指令不允许”。

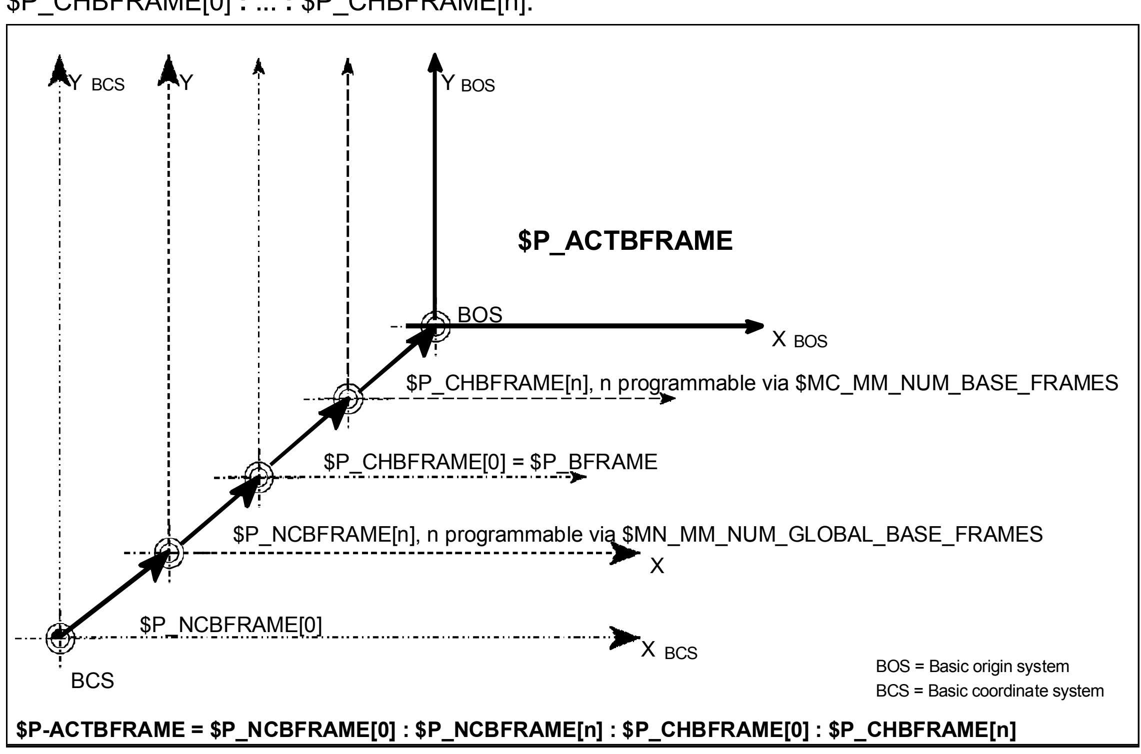
$P_ACTBFRAME

总的基准框架

变量 $P_ACTBFRAME 求算级联的总基准框架。

该变量仅可读。

$P_ACTBFRAME 等同于
$P_NCBFRAME[0] **:** ... **:** $P_NCBFRAME[n] **:**
$P_CHBFRAME[0] **:** ... **:** $P_CHBFRAME[n].





$P_CHBFRMASK 和 $P_NCBFRMASK

总的基准框架

通过系统变量 $P_CHBFRMASK 和 $P_NCBFRMASK 用户可以选择在计算"总的"基准框架时包含哪几个基准框架。变量仅在程序中编程，通过机床控制面板读入。变量的值翻译为位的掩码，说明$P_ACTBFAME的哪个基准框架数组单元进入到计算中。

使用$P_CHBFRMASK可以规定计算哪一个通道专用的基准框架，使用$P_NCBFRMASK规定哪一个NCU全局基准框架计算在内。

编程这些变量重新计算总的基准框架和总的框架。复位之后，在标准设置中有以下的数值：

$P_CHBFRMASK = $MC_CHBFRAME_RESET_MASK 和
$P_NCBFRMASK = $MN_NCBFRAME_RESET_MASK.

比如

```
$P_NCBFRMASK = 'H81'          ; $P_NCBFRAME[0] : $P_NCBFRAME[7]
$P_CHBFRMASK = 'H11'          ; $P_CHBFRAME[0] : $P_CHBFRAME[4]
```

$P_IFRAME

当前可设定的框架

通过预定义的框架变量 $P_IFRAME 可以在零件程序中读写当前可设定的框架，它是通道中有效的框架。写入的可设定框架立即计算在内。
在NCU全局的、可设定的框架中，修改的框架仅在编程的通道中生效。如果要求修改一个NCU所有通道的框架，则必须同时说明 $P_UIFR[n] 和 $P_IFRAME。然后，其它的通道还必须激活比如带G54的框架。

自软件版本**SW6.1**起
当前的系统框架，用于
$P_TOOLFRAME TOROT 和 **TOFRAME**
自软件版本**SW6.3**起
$P_WPFRAME 工件基准点
自软件版本**SW6.4**起
$P_TRAFRAME 转换

通过这些系统变量可以在零件程序中读写当前的系统框架。

$P_PFRAME
当前可编程的框架
$P_PFRAME 是一个编程的框架, 它由编程
TRANS/ATRANS, G58/G59, ROT/AROT,
SCALE/ASCALE, MIRROR/AMIRROR
产生，或者通过给可编程的框架FRAME 赋值
CTRANS, CROT, CMIRROR, CSCALE 而生成。
当前可编程的框架变量，建立可设定零点坐标系
(ENS)和工件坐标系(WKS)之间的关系。

自软件版本**SW6.3**起
当前的系统框架，用于
$P_CYCFRAME 循环
通过这些系统变量可以在零件程序中读写当前的系统框架。

$P_ACTFRAME

当前的总框架

当前生成的总框架 $P_ACTFRAME

由所有基准框架、当前可设定框架和可编程框架级联

而成。如果框架分量改变，则当前框架会更新。

$P_ACTFRAME 等同于（自软件版本SW 6.3起）：

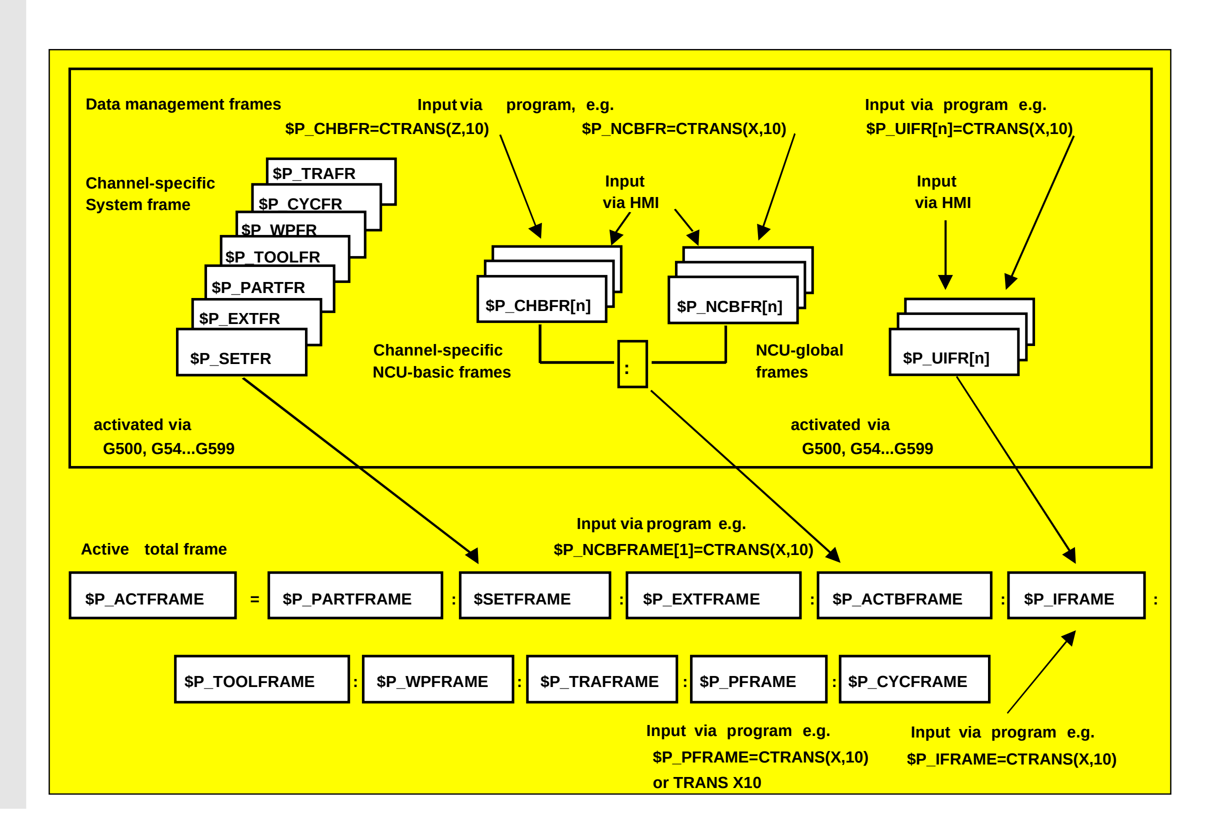
$P_SETFRAME : $P_EXTFRAME : $P_PARTFRAME : $P_ACTBFRAME :

$P_IFRAME : $P_TOOLFRAME : $P_WPFRAME : $P_PFRAME : $P_CYCFRAME

$P_ACTFRAME 等同于（自软件版本SW 6.4起）：

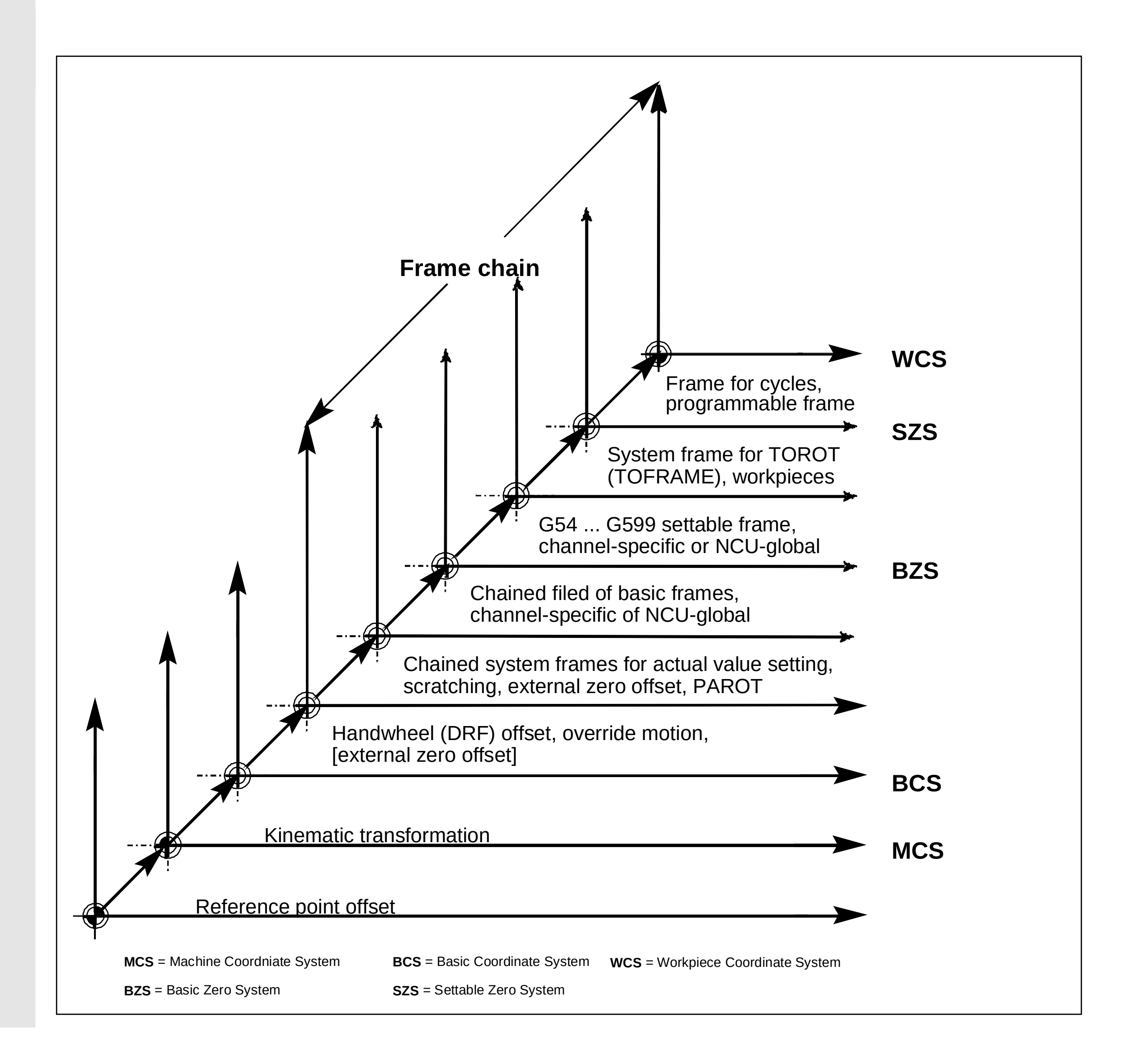
$P_PARTFRAME : $P_SETFRAME : $P_EXTFRAME: $P_ACTBFRAME : $P_IFRAME :

$P_TOOLFRAME : $P_WPFRAME : $P_TRAFRAME: $P_PFRAME : $P_CYCFRAME



框架级联

根据以上所说的当前的总框架，当前的框架由总的基准框架、可设定的框架、系统框架和可编程的框架组成。

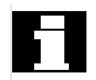


■

用于记录

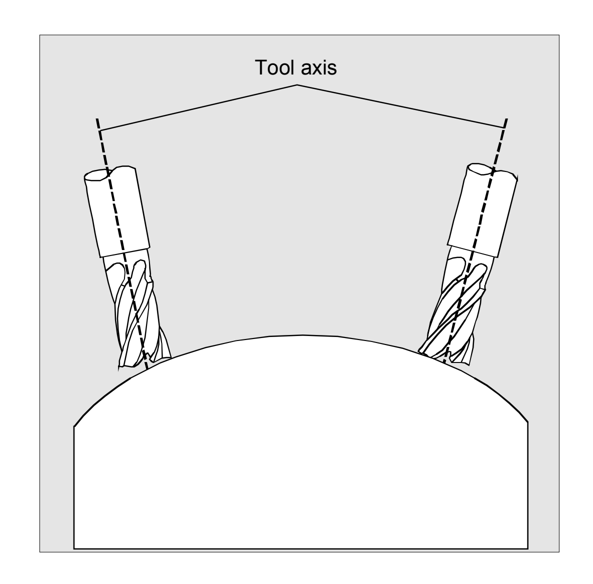
转换

7.1 三轴，四轴和五轴转换： TRAORI



为了在空间加工曲线面积的时候达到最佳切削条件，刀具的后角必须可以改变。

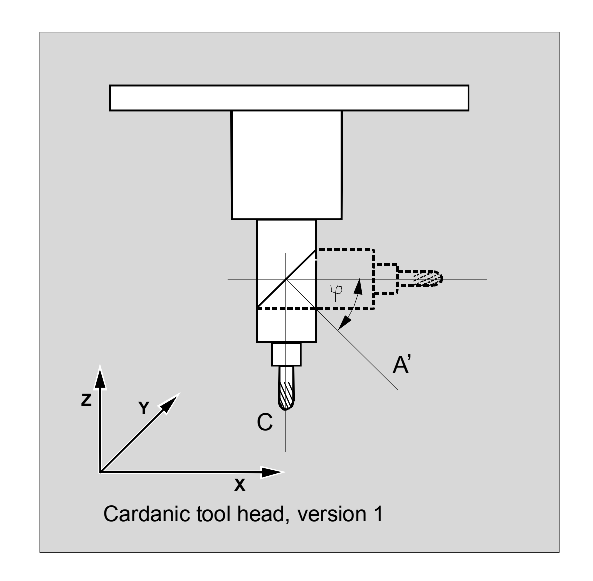
用哪一种机床结构达到这一点，这存储在轴数据中。



万向切削头

这里三个线性轴（X，Y，Z）以及两个定向轴确定了后角和刀具的工作点。两个定向轴的其中之一是作为斜置轴设置的，在这里为A'
（在很多情况下是 45°）。

定向轴的轴顺序和刀具的运动方向取决于通过机床数据的机床类型来设置。在这里给出的例子中可以看到机床类型CA的布置！



Cardanic tool head, version 1

适用于下面的关系：

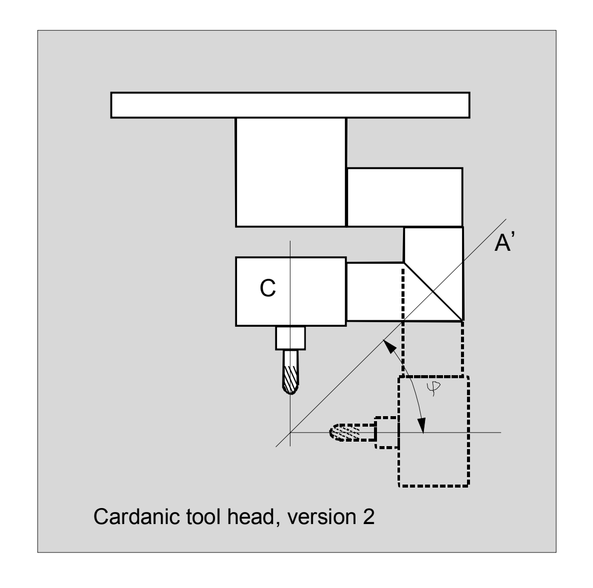
A' 和X轴呈角度

B' 和Y轴呈角度

C' 和Z轴呈角度

角度可以在范围0°到+89°通过机床数据来设计。

在NC程序中，当前有效的工作平面（G17，G18，G19）取决于所选择的刀具运动方向来设置，以便刀具长度补偿在刀具运动方向上有效。



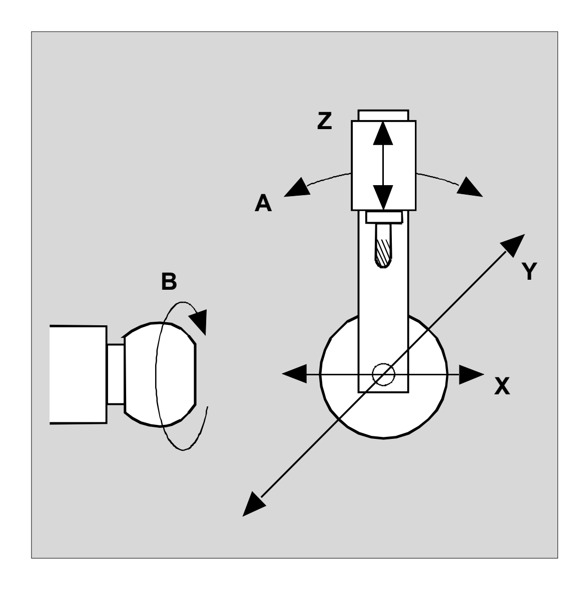
Cardanic tool head, version 2

带可回转的线性轴的转换

这种情况与运动的工件和运动的刀具的布置有关。运动学由三个线性轴(X，Y，Z)和两个直角布置的旋转轴组成。第一个回转轴例如可以通过两个线性轴的一个十字滑板来运动，刀具和第三个线性轴平行。第二个旋转轴使工件旋转。第三个线性轴（转向轴）在十字滑板的平面上。

回转轴的轴顺序和刀具的运动方向通过机床数据设置，与机床运动类型有关。

适用于下面的关系：



轴：	轴顺序：					
1. 回转轴	A	A	B	B	C	C
2. 回转轴	B	C	A	C	A	B
转向的线性轴	Z	Y	Z	X	Y	X

3轴和4轴的转换

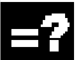
3轴和4轴的转换是5轴转换的特殊形式。
用户可以设计两个或者三个移动的轴和一个回转的轴。
转换以回转轴与定向平面正交为前提。
刀具只在与回转轴垂直的平面上才可以定向。转换支
持带有运动刀具和运动工件的机床类型。

3轴和4轴转换的设计和编程与5轴的转换类似。



编程

```
TRAORI(n)
TRAORI(n,X,Y,Z,A,B)
TRAFOOF
```



指令说明

TRAORI	激活第一个约定的方向转换
TRAORI(n)	激活用n约定的方向转换
n	转换的编号（n = 1或者2），TRAORI（1）与方向转换一相同
X,Y,Z	刀具显示出的定向矢量的组成部分
A,B	回转轴的可编程的偏移（SW 7.1及更高版本）
TRAFOOF	解除转换



其他说明

在接通转换之后位置说明（X，Y，Z）总是和刀尖有关。参与转换的回转轴位置的改变导致其他加工轴的补偿运动，刀尖位置保持不变。
定向转换总是从刀尖指向刀具安装位置。

转换的属性举例：
刀具的基本方向显示在：

TRAORI(1,0,0,1) Z方向
TRAORI(1,0,1,0) Y方向
TRAORI(1,0,1,1) Y/Z方向
（和位置45°相对应）

定向轴的偏移（SW 7.1及更高版本）
在激活定向转换时还可以直接编程一个定向轴的附加偏移。
如果在编程时保持正确的顺序不变，参数可以不设。

举例
TRAORI(, , , , A,B) 如果只要输入唯一的一个偏移。

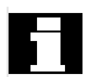
除了直接编程之外，定向轴的附加偏移还可以自动接收目前有效的零点偏移。接收通过机床数据来设计。



参考文献

/FB/ F2，3至5轴转换

7.1.1 刀具定向编程

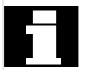


通常情况下5轴程序由CAD/CAM系统生成并且没有输入到控制系统。所以下列解释主要是针对后置处理器的程序员。

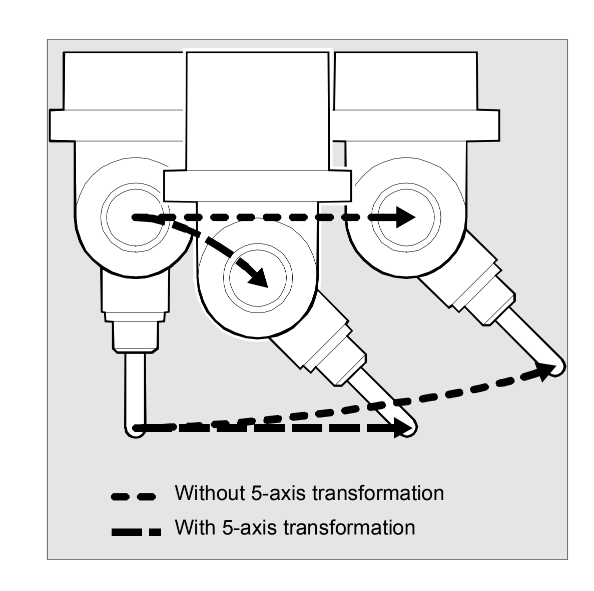
对于刀具的定向编程有3个可能：

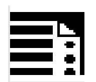
1. 直接编程回转轴运动。定向改变总是在基准坐标系或者机床坐标系统中发生。定向轴作为同步轴运行。
2. 通过A2，B2，C2编程欧拉角或者RPY角
   或者
   通过A3，B3，C3编程方向矢量。方向矢量从刀尖指向刀具安装位置。
3. 通过超前角LEAD和侧向角TILT（端面铣）编程。

在所有的情况下，只有当一个定向转换生效时，定向编程才是允许的。



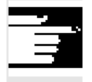
优点：这些程序在每个机床运动类型都是可传送的。



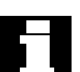


编程

G1 X Y Z A B C	编程回转轴运动
G1 X Y Z A2 = B2= C2=	编程欧拉角
G1 X Y Z A3= B3== C3==	编程方向矢量
G1 X Y Z A4== B4== C4==	在程序段开始时编程面正交矢量
G1 X Y Z A5= B5== C5==	在程序段结束时编程面正交矢量
LEA=D	刀具定向编程的超前角
TILT=	刀具定向编程的侧向角



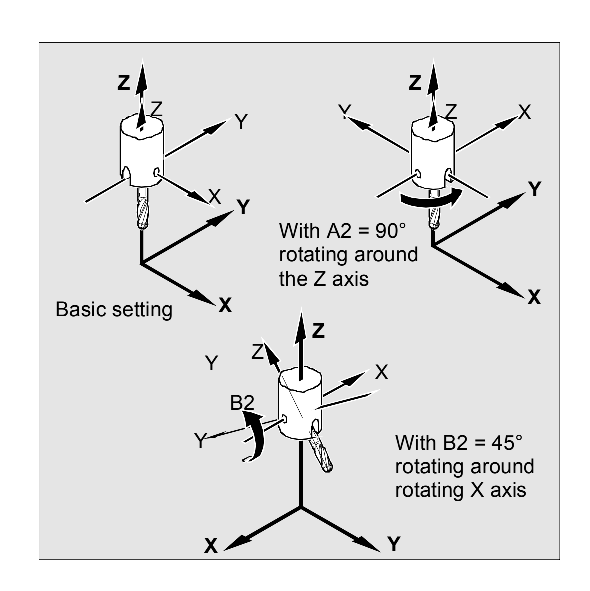
通过机床数据可以在欧拉角和RPY角之间转换。



编程欧拉角

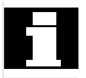
在定向编程时用A2，B2，C2编程的值被解释为欧拉角（以度来表示）。

通过一个在Z方向的矢量首先用A2围绕Z轴，然后用B2围绕新的X轴并且最后用C2围绕新的Z轴旋转，产生方向矢量。



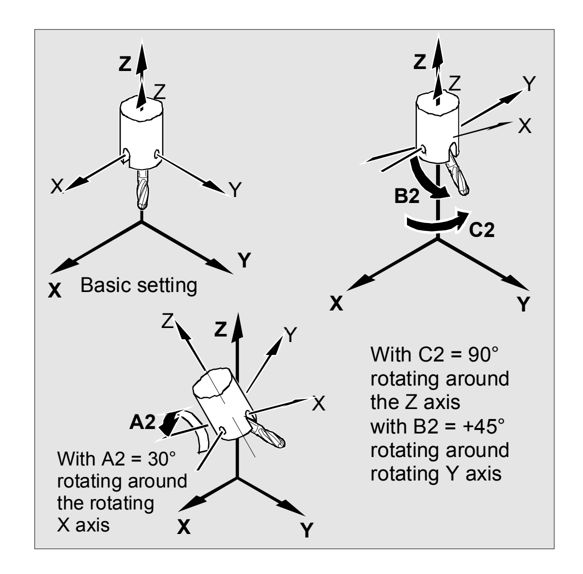


在这种情况下C2（围绕新的Z轴旋转）的值没有意义并且不需要编程。



编程RPY角

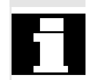
在定向编程时用A2，B2，C2编程的值被解释为RPY角（以度来表示）。



通过一个在Z方向的矢量首先用C2围绕Z轴，然后用B2围绕新的Y轴并且最后用A2围绕新的X轴旋转，产生方向矢量。



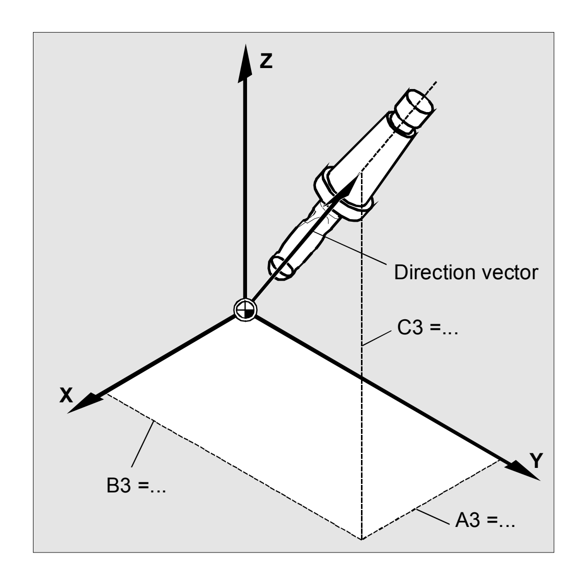
和欧拉角编程相反这里所有的三个值对方向矢量都有影响。



编程方向矢量

方向矢量的组成部分用A3，B3，C3来编程。矢量显示在刀具安装方向；矢量的长度在此没有意义。

没有编程的矢量组成部分设置为零。

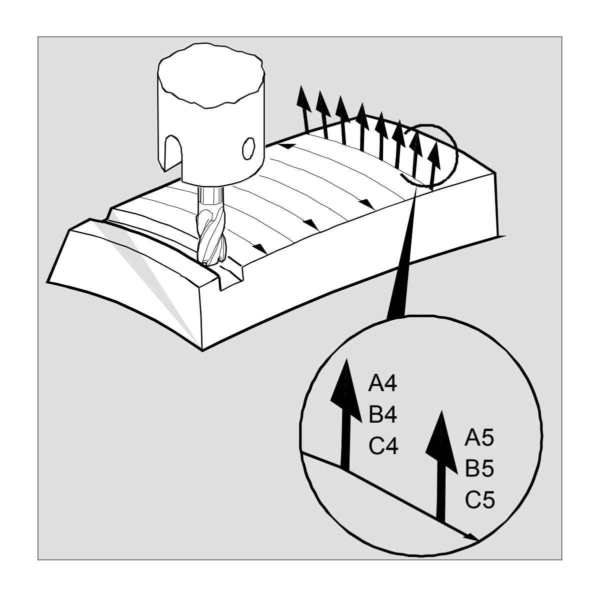


端面铣

端面铣用来加工任意弯曲的表面。

对于这种3D铣削的类型，您需要在工件表面上给3D轨迹按行描述。
计算要考虑到刀具类型和刀具尺寸，通常在CAM里执行。
计算完成的NC程序段然后通过后置处理器读取到控制器。

面描述



轨迹弯曲通过面正交矢量用下列组成部分来描述：
A4，B4，C4在程序段开始处的起始矢量
A5，B5，C5在程序段结束处的结束矢量

如果在一个程序段中只有起始矢量，那么整个程序段的面正交矢量是恒定不变的。

如果在一个程序段中只有结束矢量，那么从前面程序段的结束值到编程的结束值通过大圆插补来插补。

如果编程起始矢量和结束矢量，那么在这两个方向之间同样通过大圆插补来插补。因此生成连续的平滑的轨迹行程。

在基本设定中，在Z方向显示面正交矢量（独立于当前有效平面G17至G19）。

矢量的长度没有意义。

没有编程的矢量组成部分设置为零。
在ORIWKS当前有效时（参见以下几页）面正交矢量和当前有效的框架有关，并且随着框架旋转而旋转。

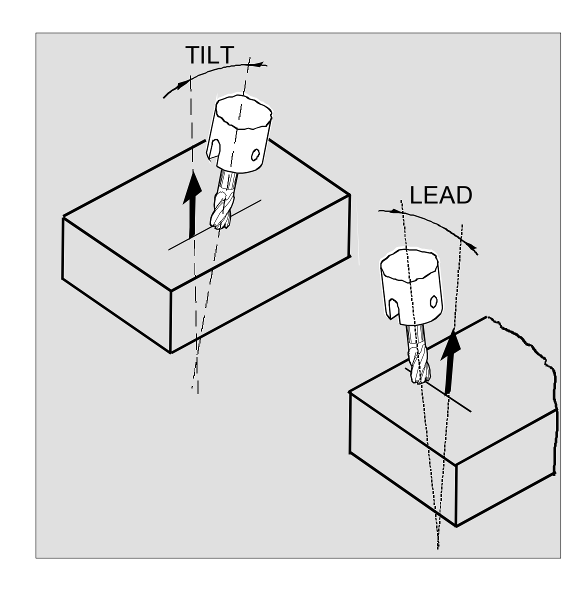
面正交矢量必须在一个通过机床数据设置的极限值内
垂直于轨迹切线，否则会报警。

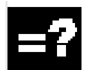


用LEAD和TILT编程刀具定向

生成的刀具定向通过以下几项得出：

– 轨迹切线，
– 面正交矢量
– 超前角LEAD
– 在程序段结束处的侧向角TILT





指令说明

`LEAD`	相对于面正交矢量的角度，在由轨迹切线和面正交矢量展开的平面上
`TILT`	平面上的角度，相对于面正交矢量垂直于轨迹切线

内角的特性（在3D-WZK时）

如果缩短内角处的程序段，那么生成的刀具定向同样
在程序段结束处可以达到。



7.1.2 刀具定向的旋转：ORIROTA, ORIROTR, ORIROTT

编程

N.. ORIROTA 或者 N.. ORIROTR 或者 N.. ORIROTT	确定旋转矢量的插补 （SW 6.1及更高版本）
THETA	激活方向矢量的旋转
THETA=<值>	以度表示的旋转角度，这个角度在程序段结束时达到
THETA=Θ_e	带旋转矢量终端角度Θ_e 的旋转角度
THETA=AC(....)	程序段方式转换到绝对尺寸说明
THETA=IC(....)	程序段方式转换到相对尺寸说明
PO[THETA]=(d_2, d_3, d_4, d_5)	用5级多项式插补旋转角度

指令说明

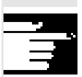
ORIROTA	对于一个绝对规定的旋转方向的旋转角度
ORIROTR	相对于平面在起始方向和结束方向之间的旋转角度
ORIROTT	相对于方向矢量改变的旋转角度
THETA	方向矢量的旋转
Θe	旋转矢量的结束角度，绝对的用G90，相对的用G91（相对尺寸）均有效。
PO[THETA]=(....)	旋转角度的多项式



功能

如果在带运动刀具的机床类型中，要求刀具的方向也可以改变，那么每个程序段均要编程结束方向。取决于机床类型可以编程定向轴的运动方向或者编程方向矢量THETA的旋转方向。对于这些旋转矢量可以编程不同的插补类型：

- ORIROTA 对于一个绝对规定的旋转方向的旋转角度。
- ORIROTR 相对于起始方向和结束方向平面的旋转角度。
- ORIROTT 相对于方向矢量改变的旋转角度。



带定向轴的刀具方向在后面的章节中描述。

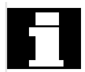


工作流程

ORIROTA
旋转角度THETA根据一个绝对确定的空间方向来插补。基本旋转方向通过机床数据生成。

ORIROTR
旋转角度THETA相对于平面插补，平面由起始方向和结束方向展开。

ORIROTT
旋转角度THETA相对于方向改变插补。只有至少编程一个“倾斜角PSI”的多项式用于方向，并且THETA=0时对旋转矢量进行切线插补，用于方向改变，则它与ORIROTR不同。因此生成一个不在平面上运行的方向改变。通过一个附加编程的旋转角度THETA可以例如：插补旋转矢量，使矢量始终构成一个确定的值，用于方向改变。



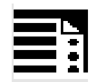
只有当插补类型ORIROTA当前有效时，旋转角度或者旋转矢量可以以四种可能的类型如下编程：

1. 直接的回转轴位置A，B，C
2. 通过A2，B2，C2的欧拉角（以度表示）
3. 通过A2，B2，C2的RPY角（以度表示）
4. 通过A3，B3，C3的方向矢量（旋转角度借助THETA=<值>）。

如果ORIROTR或者ORIROTT有效，那么旋转角度只可以直接用THETA编程。

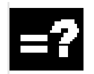
旋转也可以单独在一个程序段中编程，不需要进行定向改变。在此ORIROTR和ORIROTT没有意义。在这种情况下旋转角度总是根据绝对方向插补（ORIROTA）。

7.1.3 定向轴的关系 ORIWKS, ORIMKS



编程

```
N.. ORIMKS=
```
或者
```
N.. ORIWKS=
```



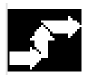
指令说明

ORIMKS	机床坐标系中的旋转
ORIWKS	工件坐标系中的旋转



功能

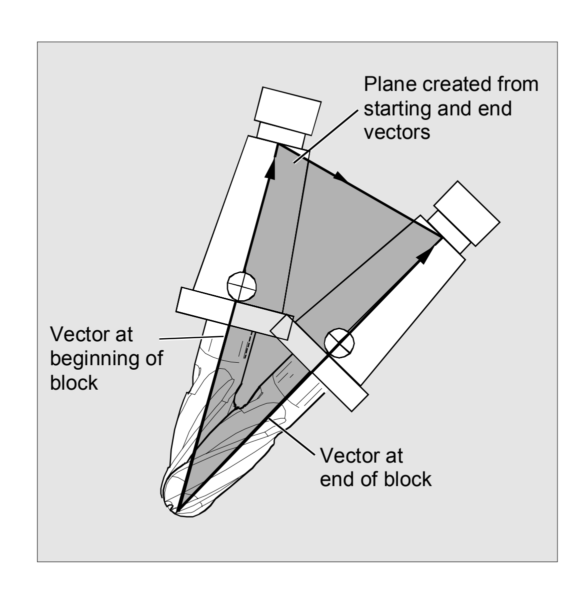
在用欧拉角和RPY角或者方向矢量在工件坐标系中定向编程时，旋转运行的过程可以通过ORIMKS/ORIWKS来设置。

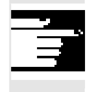


工作流程

在使用ORIMKS时设计的刀具运行取决于机床类型。用空间固定的刀尖改变方向时在回转轴位置之间进行线性插补。

在使用ORIWKS时刀具运行不取决于机床类型。用空间固定的刀尖改变方向时刀具在由起始矢量和终点矢量展开的平面上运行。





其他说明

ORIWKS是缺省设置。

如果一个5轴程序开始不清楚应在哪个机床上展开，那么原则上必须选择ORIWKS。机床实际上执行的运行取决于机床类型。

用ORIMKS 可以编程实际的机床运行，
用来避免例如：和辅助设备的碰撞或诸如此类的情况。

用机床数据$MC_ORI_IPO_WITH_G_CODE可以确定
哪种插补类型有效：
ORIMKS/ORIWKS或者ORIMACHAX/ORIVIRTAX(参见章节7.1.4)。

7.1.4 单个的位置和它的操作



对ORIWKS的说明：

在5轴机床的单配置范围内的定向运行要求加工轴的大幅度运行。（例如在使用回转头时用C作为旋转轴并且A作为摆动轴A = 0所有位置都是单个的。）

为了不使加工轴过载，速度导向将单个位置附近的轨迹速度大幅减小。

用机床数据
`$MC_TRAFO5_NON_POLE_LIMIT`
`$MC_TRAFO5_POLE_LIMIT`
可以给定转换的参数，以便在极点附近的定向运动通过极点来进行并且可以顺利地进行加工。



对软件版本SW5.2的说明：

自软件版本SW5.2起单个的位置只用MD
`$MC_TRAFO5_POLE_LIMIT`来操作（参见功能描述第3部分，章节2.8.4）。

7.1.5 定向轴(SW 5.2及以上版本): ORIAXES, ORIVECT, ORIEULER, ORIRPY

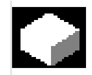
编程

```
N..ORIAXES 或者 ORIVECT                 ；定向插补：
N..G1 X Y Z  A B C                      ；线性或者大圆插补
N..ORIEULER 或者 ORIRPY                 ；定向编程：
N..G1 X Y Z A2= B2= C2=                 ；关于欧拉/RPY的定向角
或者
N..ORIVIRT1 或者 ORIVIRT2               ；关于虚拟定向轴的定向角
N..G1 X Y Z A3= B3= C3=

N.. ORIPLANE                            ；定向插补（SW 5.3及更高版本）
N.. ORICONCW 或者 ORICONCCW             ；扩展插补（SW 6.1及更高版本）
N.. ORICONTO 或者 ORICURVE              ；在一个圆锥表面上
N.. G1 X Y Z A6= B6= C6=
N.. ORICONIO
N.. G1 X Y Z A7= B7= C7=
```

指令说明

ORIAXES	加工轴或者定向轴的线性插补
ORIVECT	大圆插补（和ORIPLANE一致）
ORIMKS ORIWKS	在机床坐标系中旋转 在工件坐标系中旋转 描述参见章节7.1.2
G1 X Y Z A B C	加工轴位置编程
ORIEULER	通过欧拉角的定向编程
ORIRPY	通过RPY角的定向编程
G1 X Y Z A2= B2= C2=	虚拟轴的角度编程
ORIVIRT1 ORIVIRT2	通过虚拟定向轴的定向编程 (定义1)，根据MD $MC_ORIAX_TURN_TAB_1确定 (定义2)，根据MD $MC_ORIAX_TURN_TAB_2确定
G1 X Y Z A3= B3= C3=	方向轴的方向矢量编程
	定向插补（SW 5.3及更高版本）
ORIPLANE	平面上的插补（大圆插补）
ORICONCW	顺时针方向圆锥表面上的插补
ORICONCCW	逆时针方向圆锥表面上的插补
ORICONTO	在切线过渡的圆锥表面上的插补
G1 X Y Z A6= B6= C6=	圆锥的旋转轴的编程（标准化的矢量）
ORICONIO G1 X Y Z A7= B7= C7=	在圆锥表面上带中间定向说明的插补（作为标准化矢量编程）
ORICURVE XH YH ZH 带多项式 PO[XH]=(xe, x2, x3..)	带刀具两个接触点运行说明的定向插补除了各个终点之外空间曲线多项式可以编程。



功能

定向轴的功能描述了空间里刀具的定向。对此更进一步的引入了一个第三自由度，这个自由度描述了围绕自身的旋转。这个对6轴转换是必要的。



用MD$MC_ORI_DEF_WITH_G_CODE可以确定怎样定义编程设计的角度A2, B2, C2：
根据MD$MC_ORIENTATION_IS_EULER（标准）产生定义或者根据G_组50来产生定义
(ORIEULER, ORIRPY, ORIVIRT1, ORIVIRT2).

用MD$MC_ORI_IPO_WITH_G_CODE可以确定哪种插补类型有效：
ORIWKS/ORIMKS 或者 ORIAXES/ORIVECT.

运行方法 JOG

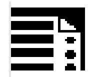
定向角在这样的运行方式下总是线性插补。在通过运行键操作的连续的，增量的运动时只能运行一个定向轴。通过手轮可以同时运行定向轴。



对于定向轴的手动运行通道特有的进给补偿开关或者快速补偿开关在快速叠加时有效。

用下列的机床数据可以有一个单独的速度说明：
$MC_JOG_VELO_RAPID_GEO
$MC_JOG_VELO_GEO
$MC_JOG_VELO_RAPID_ORI
$MC_JOG_VELO_ORI

自软件版本SW6.3起
利用手动运行功能，在SINUMERIK 840D带
“转换软件包处理”功能时，以及在
Sinumerik 810D powerline 中自 SW 6.1起，
几何轴的平移在基准坐标系MKS、WKS和TKS中可以在JOG运行方式分开设定。



编程进给

FORI1	在大圆上方向矢量摆动的进给
FORI2	围绕摆动方向矢量叠加的旋转的进给



可以编程的进给在定向运行时等于一个角速度[度/分钟]。

通过G代码进给的有效性：

在编程ORIAXES时可以通过FL[ ]-
指令（进给极限）来限制定向轴的进给。

在编程ORIVECT时进给必须用FORI1或者FORI2来编程。在NC程序段中FORI1和FORI2只允许编程一次。在这个编程时总是运行最短行程。
在旋转和摆动的叠加运行时，当时最小的进给有效。
进给在定向运行时为一个角速度[度/分钟]。

如果几何轴和定向轴的运行轨迹相同，那么运行由最小的进给决定。

扩展插补（SW 6.1及更高版本）
用扩展的定向可以沿位于空间的圆锥表面执行方向改变。需要以下的编程说明：

- 通过前面程序段的起始方向
- 终点处的方向要么通过方向矢量用A3，B3，C3，要么在欧拉角或者在RPY角度通过A2，B2，C2
- 圆锥的旋转轴作为矢量用A6, B6, C6
- 带有名称NUT的孔径角PSI
- 用A7, B7, C7的中间定向



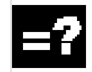
参考文献说明：
SINUMERIK 840D/FM-NC 功能描述（第3部分），
“转换软件包处理”。
/FB/ F2, 3至5轴转换

7.1.6 直角PTP-运行（自软件版本SW5.2起)



编程

```
N.. TRAORI
N.. STAT=`B10` TU=`B100` PTP
N.. CP
```



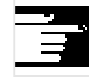
指令说明

PTP	点到点（点到点运行） 作为同步轴运动执行运行；参与运行的最慢的轴是速度主导轴。
CP	连续的轨迹（轨迹运行） 作为直角轨迹运行执行运行
STAT =	铰接的位置；值取决于转换。
TU=	TURN-信息 因此可以明确地返回到-360度和+360度之间的轴角度。

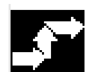


功能

用这个功能可以编程一个在直角坐标系上的位置，但是机床的运行在机床坐标系中完成。
如果是通过单一性来运行的，这个功能可以在更换铰接位置时应用。



提示：
这个功能只有与有效转换相联系时才有意义。此外“PTP-运行”只有与G0和G1联系时才允许。



工作流程

在直角运行和加工轴运行之间的转换通过指令PTP和CP来完成。这是模态有效的。CP是标准设置。

位置编程(STAT=)

机床设置不只由带直角坐标的位置说明和刀具的方向决定。分别根据不同的机床类型，存在最多8种不同的或者有区别的铰接配置。这些是转换专用的。为了将一个直角位置明确地换算成轴角度，铰接的位置必须用指令STAT=说明。指令“STAT”包含一个位作为二进制值，用于每个可能的位置。



参考文献说明：

不同的转换包含在手册中：
SINUMERIK 840D/FM-NC
功能描述（第3部分），“转换软件包处理”。

必须在“STAT”编程的位置位包含在手册：
SINUMERIK 840D/FM-NC
功能描述（第3部分），“3至5轴转换”。

轴角度编程(TU=)

为了可以明确地返回到轴角度
< ±360，必须用指令“TU=”编程该信息。
这个指令是程序段方式有效。

轴以最短行程运行：

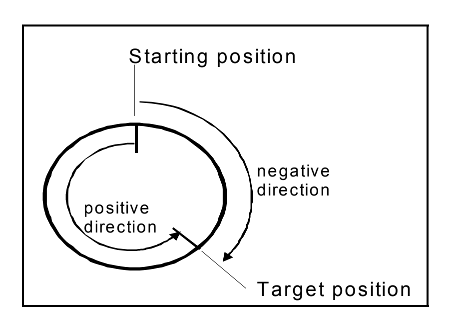
- 如果在一个位置上没有编程TU
- 在运行范围> ±360度的轴上

举例：

图中给出的目标位置可以从正向或者负向返回运行。在地址A1下编程编程这个方向。

A1=225°, TU=Bit 0, → 正向

A1=–135°, TU=Bit 1, → 负向



在CP和PTP运行之间精磨

在程序段之间可以用G641编程中间过渡磨削。
磨削范围的大小是以毫米或者英寸表示的轨迹行程，
在这个范围内进行程序段过渡磨削。大小说明如下：

- 对于G0程序段用ADISPOS
- 对于所有其他行程指令用ADIS

轨迹行程计算在非G0程序段时与对F地址的考虑相一致。在以FGROUP(..)说明的轴上执行此进给。

进给计算：
对于CP程序段，使用基准坐标系的直角轴来计算。
对于PTP程序段，机床坐标系的相应的轴用来计算。

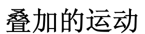


其它说明

运行方式转换
功能“直角的PTP运行”只在运行方式AUTO和MDA时有意义。在JOG之后转换运行方式时，当前设置保持不变。
如果设置了G代码PTP，那么轴在MKS中运行。如果设置了G代码CP，那么轴在WKS中运行。

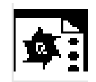
电源 开/复位
在打开电源或者复位之后，设置取决于机床数据
`$MC_GCODE_RESET_VALUES[48]`。运行类型
“CP”是按照标准设置的。

重新定位
如果在中断程序段期间设置功能“直角PTP运行”，
那么也可以用PTP返回定位。

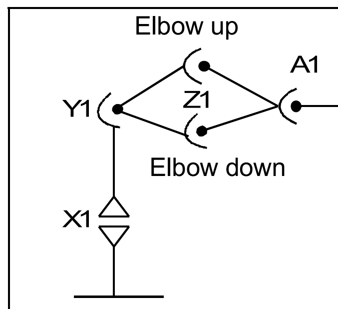


叠加的运动

DRF偏移或者外部的零点偏移仅在直角PTP运行时被限制。在从PTP－到CP－运行的转换时在BKS中不允许有叠加。



编程举例



N10	G0 X0 Y-30 Z60 A-30 F10000	起始位置 → 肘关节上部
N20	TRAORI(1)	转换 开
N30	X1000 Y0 Z400 A0	
N40	X1000 Z500 A0 STAT=´B10´ TU=´B100´ PTP	没有转换，重新定向 → 肘关节下部
N50	X1200 Z400 CP	转换再次有效
N60	X1000 Z500 A20	
N70	M30	

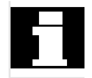
7.1.7 在线刀具长度补偿：TOFFON, TOFFOF (自软件版本SW 6.4起)

编程

```
N.. TRAORI
N.. TOFFON(X,25)
N.. WHEN TRUE DO $AA_TOFF[X]
```

指令说明

TOFFON	刀具偏移 开（激活在线－刀具长度补偿） 在激活时可以对于相应的补偿方向说明一个偏移值，这个值立刻可以得出。
TOFFOF	刀具偏移 关（复位在线－刀具长度补偿） 相应的补偿值被复位并且释放进刀停。
X, Y, Z,	说明的偏移值的补偿方向



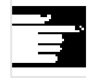
提示：
在线－刀具长度补偿是一个选项，这个选项之前必须释放。只有在和当前有效的定向转换或者和一个当前有效的可定向刀架相联系，这个功能才有效。



功能

通过这个系统变量$AA_TOFF[]可以实时三维叠加和三个刀具方向相符的有效的刀具长度。
三个几何轴命名符作为变址使用。与此同时当前有效的补偿方向的数量由同时有效的几何轴来确定。
所有的补偿可以同时有效。

应用
功能在线－刀具长度补偿可以在以下情况应用：
- 定向转换TRAORI
- 可定向的刀架TCARR



其它说明

程序段预处理

在运行程序段预处理时要考虑到在主运行中有效的当前刀具长度偏移。为了进一步充分利用允许的最大轴速度，需要用一个进刀停STOPRE，在建立刀具偏移这段时间里来暂停程序段预处理。

如果刀具长度补偿在程序启动之后不再改变，或者如果在改变刀具长度补偿之后执行多个程序段，它多于进刀运行和主运行之间IPO－缓冲器所能接收地程序段，那么刀具偏移在进刀时也始终已知。

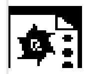
变量$AA_TOFF_PREP_DIFF

当前插补器中有效的补偿，与在程序段预处理时有效的补偿之间的差值尺寸，可以在变量$AA_TOFF_PREP_DIFF[ ]中询问。

设置机床数据和设定数据

对于在线－刀具长度补偿可以使用以下机床数据：

- MD 20610:ADD_MOVE_ACCEL_RESERVE
  速度曲线的备用
- MD 21190:TOFF_MODE 系统变量的内容
  $AA_TOFF[ ]作为绝对值得出或者求积分。
- MD 21194:TOFF_VELO
  在线－刀具长度补偿的速度
- MD 21196:TOFF_ACCEL
  在线－刀具长度补偿的加速度
- 说明极限值的设定数据SD 42970:TOFF_LIMIT
  刀具长度补偿值的上限



编程举例

选择刀具长度补偿

```
MD 21190: TOFF_MODE          =1          ；返回绝对值
MD 21194: TOFF_VELO[0]       =1000
MD 21196: TOFF_VELO[1]       =1000
MD 21194: TOFF_VELO[2]       =1000
MD 21196: TOFF_ACCEL[0]      =1
MD 21196: TOFF_ACCEL[1]      =1
MD 21196: TOFF_ACCEL[2]      =1
```

```
N5    DEF REAL XOFFSET
N10   TRAORI(1)                               ；转换 开
N20   TOFFON(Z)                               ；激活在线－WZL－补偿；对于Z－刀具
                                              ；方向
N30   WHEN TRUE DO $AA_TOFF[Z] = 10           ；用于Z－刀具方向插补一个WZL－补偿
      G4 F5                                   ；10
...
N40   TOFFON(X)                               ；激活在线－WZL－补偿；用于X－刀具方
                                              ；向
N50   ID=1 DO $AA_TOFF[X] = $AA_IW[X2]        ;用于X－刀具方向取决于轴X2的位置执行
      G4 F5                                   ；一个补偿
...
N100  XOFFSET = $AA_TOFF_VAL[X]               ；赋值当前在X方向的补偿
N120  TOFFON(X, -XOFFSET)                     ；对于X－刀具方向WZL－补偿再次返回到0
      G4 F5
```

选择刀具长度补偿

```
N10   TRAORI(1)                               ；转换 开
N20   TOFFON(X)                               ；激活Z－刀具方向
N30   WHEN TRUE DO $AA_TOFF[X] = 10           ；对于X－刀具方向插补一个WZL－补偿
      G4 F5                                   ；10
...
N80   TOFFOF(X)                               ；删除X－刀具方向的位置偏移：
                                              ；...$AA_TOFF[X] = 0
                                              ；不运行轴
                                              ；WKS中对于当前位置附加计算与当前方向
                                              ；相符的位置偏移
```



参考文献

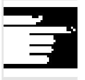
/FB/ F2, 3至5轴转换

7.2 车削件的铣削加工： TRANSMIT



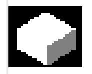
编程

```
TRANSMIT 或者 TRANSMIT(n)
TRAFOOF
```

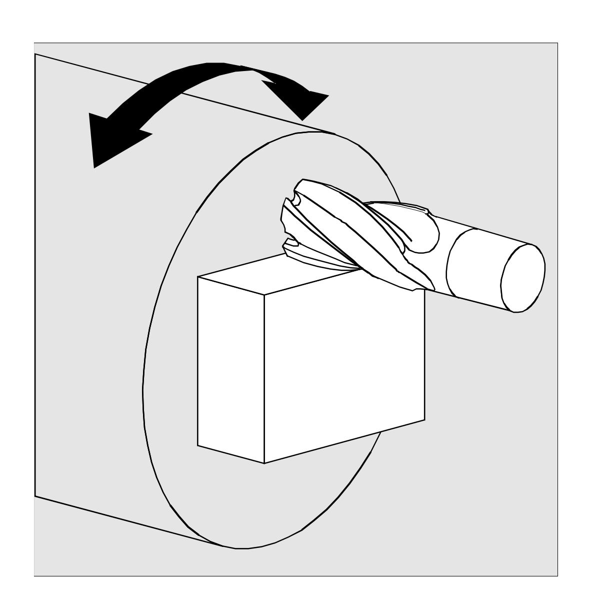


指令说明

TRANSMIT	激活第一个约定的TRANSMIT功能
TRANSMIT(n)	激活第n个约定的TRANSMIT功能；n最大可以是2(TRANSMIT(1)和TRANSMIT相符)。
TRAFOOF	关闭一个当前有效的转换



如果在每个通道中激活剩下的转换中的一个，
（例如：TRACYL，TRAANG，TRAORI），那么同
样关闭一个当前有效的转换TRANSMIT 。



功能TRANSMIT可以用于以下操作：

- 在车削中的车削件的端面加工（钻孔，轮廓）。
- 对于加工编程可以使用一个直角坐标系。
- 控制系统将编程设计的直角坐标系的运行转换成实际的加工轴的运行（标准情况）：
  – 回转轴
  – 垂直于旋转轴的横向进给轴
  – 平行于旋转轴的纵向轴
  线性轴互相垂直。
- 运行相对于旋转中心的刀具中心偏移。
- 速度导向考虑到为旋转运行所定义的限制。

回转轴

不能编程回转轴，因为它们被一个几何轴覆盖并且因此作为通道轴不能直接编程。

极点

至软件版本**SW 3.x**止

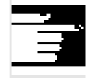
禁止通过极点（直角坐标系的坐标原点）的运行。一个通过极点的运行会停止在极点并且发出报警。在铣刀中心偏移时，运行停止在不可返回范围的边缘。

自软件版本SW 4起

对于极点运行有2种可能性：

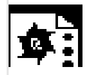
1. 线性轴单独运行
2. 以极点的回转轴旋转运行到极点并且再从极点运行出来

通过MD 24911和24951来选择。

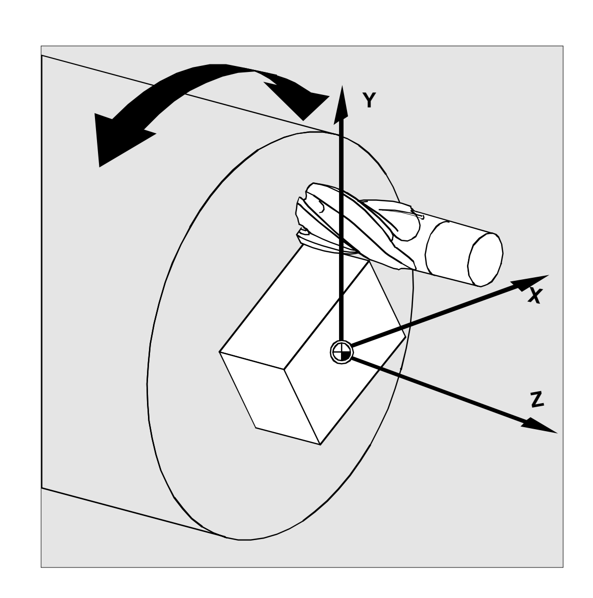


参考文献

/FB/ M1 运动学的转换

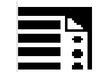


编程举例



N10 T1 D1 G54 G17 G90 F5000 G94	选择刀具
N20 G0 X20 Z10 SPOS=45	返回起始位置
N30 TRANSMIT	激活TRANSMIT-功能
N40 ROT RPL=-45 N50 ATRANS X-2 Y10	设置框架
N60 G1 X10 Y-10 G41 OFFN=1 N70 X-10 N80 Y10 N90 X10 N100 Y-10	方头粗加工；加工余量1毫米
N110 G0 Z20 G40 OFFN=0 N120 T2 D1 X15 Y-15 N130 Z10 G41	换刀
N140 G1 X10 Y-10 N150 X-10 N160 Y10 N170 X10 N180 Y-10	方头精加工
N190 Z20 G40 N200 TRANS N210 TRAFOOF	取消框架
N220 G0 X20 Z10 SPOS=45	返回起始位置
N230 M30	

7.3 圆柱表面转换： TRACYL



编程

```
TRACYL(d) 或者 TRACYL(d,t)
TRAFOOF
```



指令说明

TRACYL(d)	激活第一个在通道机床数据中约定的TRACYL功能。 d 参数用于加工直径。
TRACYL (d,n)	激活第n个在通道机床数据中约定的TRACYL功能；n最大可以是2， TRACYL(d,1)和TRACYL(d)相符。
d	加工直径的值。加工直径是刀尖和旋转中心之间的两倍距离。
TRAFOOF	转换 关（BKS和MKS再次一致）。
OFFN	偏移 轮廓－标准：编程设计的基准轮廓的槽壁的距离

如果在每个通道中激活剩下的转换中的一个，
（例如：TRANSMIT，TRAANG，TRAORI），那么
同样关闭一个当前有效的转换TRACYL。



功能

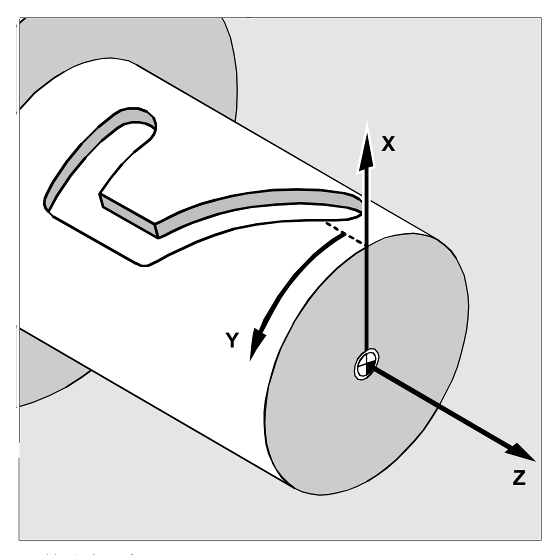
圆柱表面曲线转换 TRACYL

圆柱表面曲线转换TRACYL可以用于以下操作：

加工

- 圆柱体上的纵向槽，
- 圆柱体上的横向槽，
- 圆柱体上任意运行的槽。

槽的运行要根据展开的平面的圆柱表面来编程。



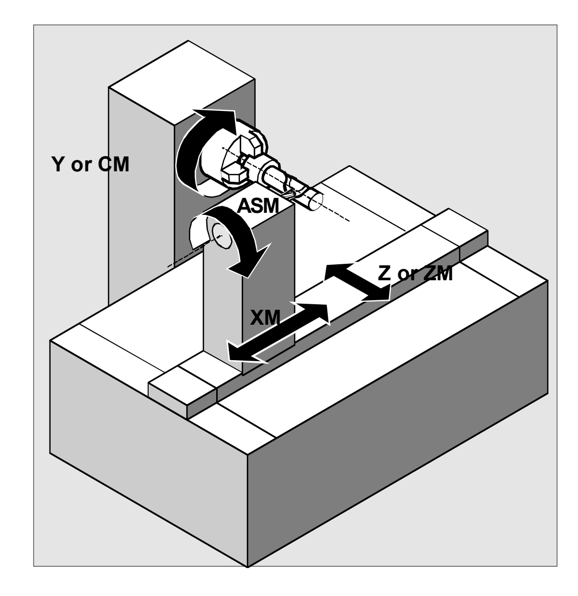
工件坐标系

在两个加工形式中有圆柱表面坐标转换：

- 不带槽壁补偿(TRAFO_TYPE_n=512)
- 带槽壁补偿(TRAFO_TYPE_n=513)

不带槽壁补偿：
控制系统将编程设计的圆柱坐标系的运行转换成实际的加工轴的运行：
– 回转轴
– 垂直于旋转轴的横向进给轴
– 平行于旋转轴的纵向轴

线性轴互相垂直。横向进给轴与回转轴相交。

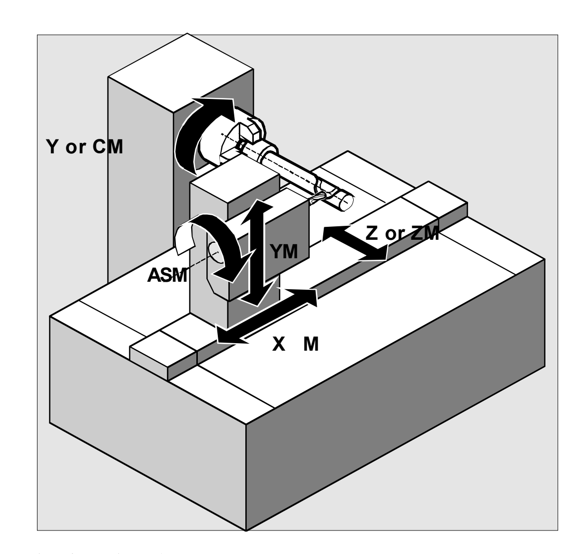


机床坐标系

带槽壁补偿：
运动类型同上，但是还有
– 平行于圆周方向的纵向轴

线性轴互相垂直。

速度导向考虑到为旋转运行所定义的限制。

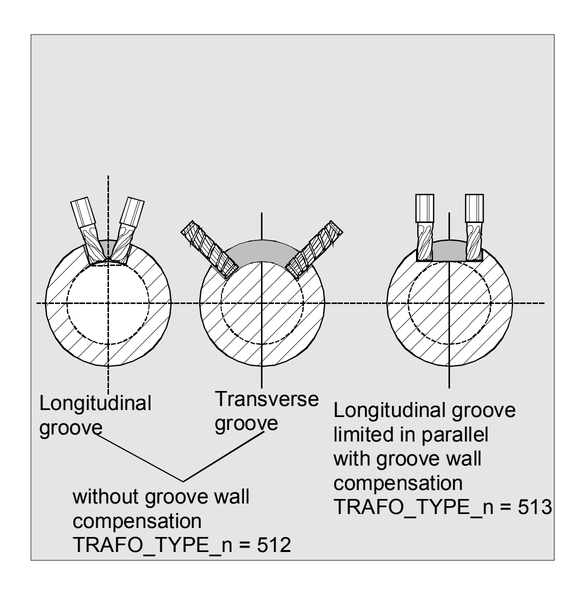


机床坐标系

槽横截面

如果槽宽正好和刀具半径相符，那么在轴配置1时，与回转轴成纵向的槽只是平行的被限制。

与圆周平行的槽（横向槽）在开始和结束时不平行。



偏移轮廓标准OFFN (513)

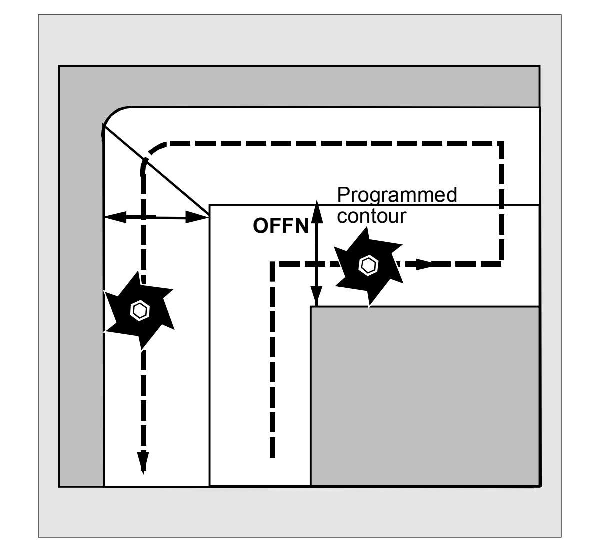
为了用TRACYL来铣削槽，可以在

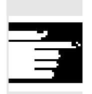
- 零件程序中编程槽中心线，
- 通过OFFN 编程**半槽宽**。

OFFN只有用选择的刀具半径补偿才有效，为了避免对槽壁的损伤。**此外OFFN>=刀具半径，为了排除对对面槽壁的损伤。**

铣削一个槽的零件程序通常由以下几个步骤组成：

1. 选择刀具
2. 选择TRACYL
3. 选择合适的坐标偏移（框架）
4. 定位
5. 编程OFFN
6. 选择WRK
7. 返回程序段（运行到WRK并且返回槽壁）
8. 槽中心线的轮廓
9. 取消WRK
10. 开始运行程序段（离开WRK并且离开槽壁）
11. 定位
12. TRAFOOF
13. 再次选择原始的坐标偏移（框架）





特点：

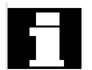
- 选择WRK：
  WRK不是根据槽壁，而是相对于编程设计的槽中心线来编程。为了使刀具在槽壁左侧运行，要输入G42（代替G41）。
  如果在OFFN里录入带负号的槽宽，就不需要这样操作了。
- 带TRACYL的OFFN和不带TRACYL的效果不同。因为OFFN不带TRACYL在当前有效的WRK时也可以计算，OFFN在TRAFOOF之后再次设置为零。
- 在零件程序内可以改变OFFN。因此槽中心线可以从中心偏移（见图）。
- 导向槽：
  用TRACYL在使用导向槽时不生成同一个槽，好像这个槽是用刀具完成的，它的直径指示出槽宽。
  原则上不可能用一个较小的圆柱形刀具生成同一个槽壁几何尺寸，用较大的刀具也不行。
  TRACYL将错误减少到最低限度。为了不出现精确性问题，刀具半径只能略小于半槽宽。



说明：

OFFN和WRK

- 在TRAFO_TYPE_n=512时值在在OFFN下作为WRK的加工余量。
- 在TRAFO_TYPE_n=513时在OFFN中编程半槽宽。轮廓用OFFN-WRK开始运行。



在带槽壁补偿的圆柱表面曲线转换时，
用于补偿的轴为零(y=0)，以便于加工与编程的槽中心
线对中的槽。

回转轴

不能编程回转轴，因为它们被一个几何轴覆盖并且因
此作为通道轴不能直接编程。

轴使用

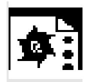
下列轴不可以作为定位轴或者摆动轴使用：

- 几何轴沿圆柱表面（Y轴）的圆周方向
- 附加的线性轴在槽壁补偿时（Z轴）

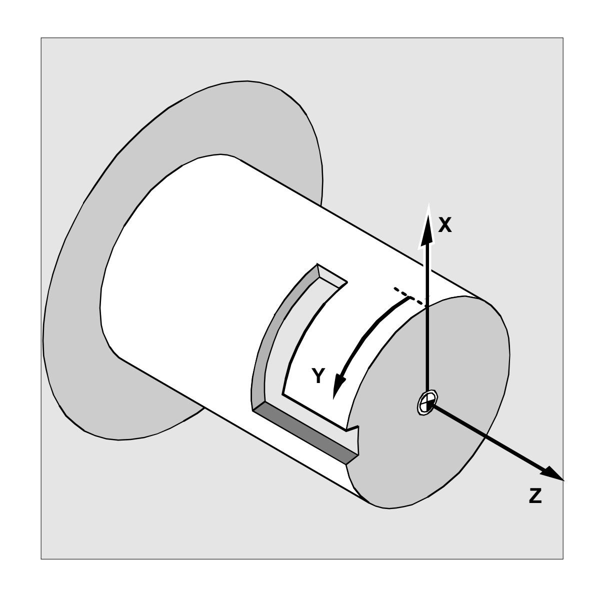
刀具的定义

以下的例子适用于测试圆柱转换TRACYL的参数确
定：

刀具参数 编号(DP)	意义	附注
$TC_DP1[1,1]=120	刀具类型	铣刀
$TC_DP2[1,1]=0	切削刃位置	只针对旋转刀具
几何尺寸	**长度补偿**	
$TC_DP3[1,1]=8.	长度补偿矢量	根据类型和平面的计算
$TC_DP4[1,1]=9.		
$TC_DP5[1,1]=7.		
几何尺寸	**半径**	
$TC_DP6[1,1]=6.	半径	刀具半径
$TC_DP7[1,1]=0	切槽锯片的槽宽b，铣削刀具的倒圆半径	
$TC_DP8[1,1]=0	超出规定范围k	只针对切槽锯片
$TC_DP9[1,1]=0		
$TC_DP10[1,1]=0		
$TC_DP11[1,1]=0	圆锥形铣削刀具角度	
磨损	**长度和半径补偿**	
$TC_DP12[1,1]=0	剩余参数直到$TC_DP24=0	基础尺寸/适配器



编程举例

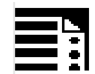


N10 T1 D1 G54 G90 F5000 G94	选择刀具，紧固补偿
N20 SPOS=0 N30 G0 X25 Y0 Z105 CC=200	返回起始位置
N40 TRACYL (40)	圆柱表面曲线转换生效
N50 G19	平面选择

加工一个吊钩形槽：

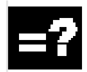
N60 G1 X20	在槽底横向进给刀具
N70 OFFN=12	确定相对于槽中心线底槽壁距离12毫米
N80 G1 Z100 G42	返回右侧槽壁
N90 G1 Z50	平行于圆柱轴的槽截面
N100 G1 Y10	平行于圆周的槽截面
N110 OFFN=4 G42	返回左侧槽壁；确定相对于槽中心线的槽壁距离4毫米
N120 G1 Y70	平行于圆周的槽截面
N130 G1 Z100	平行于圆柱轴的槽截面
N140 G1 Z105 G40	从槽壁出发
N150 G1 X25	空运转
N160 TRAFOOF	
N170 G0 X25 Y0 Z105 CC=200	返回起始位置
N180 M30	

7.4 斜置轴： TRAANG



编程

```
TRAANG(α) 或者 TRAANG(α,n)
TRAFOOF
```



指令说明

TRAANG	如果省略角度α或者输入零，那么带前面选择的参数写法的转换被激活。在第一次选择的时预设置根据机床数据。
TRAANG(α)	激活第一个约定的斜置轴转换
TRAANG(α,n)	激活第n个约定的斜置轴转换。 n最大可以是2。TRAANG(α,1)和TRAANG(α)相符。
α	斜置轴的角度
TRAFOOF	转换 关



如果省略α（角度）或者输入零，那么带前面选择的参数写法的转换被激活。在第一次选择的时预设置按照机床数据。（性能适用软件版本< 6.4，更新的版本见下）。

如果在每个通道中激活剩下转换中的一个，那么同样关闭一个当前有效的转换TRAANG。
(例如：TRACYL，TRANSMIT，TRAORI)。



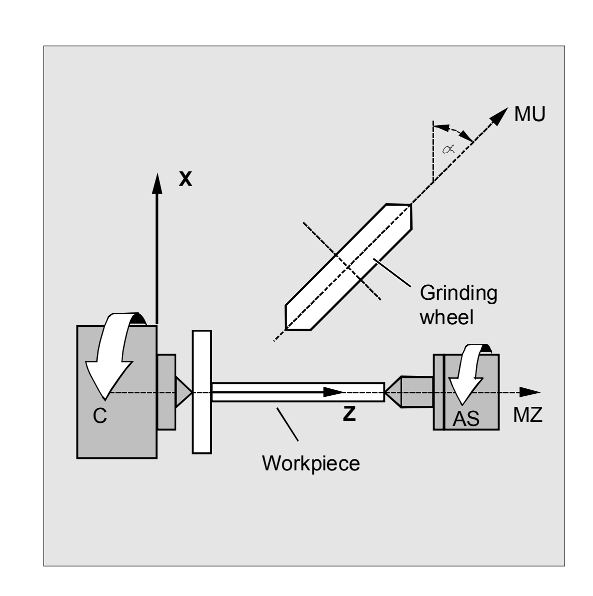
（性能适用自软件版本6.4起）
如果省略α（角度）（例如：TRAANG()，TRAANG(,n))，
那么带前面选择的参数写法的转换被激活。
在第一次选择的时预设置按照机床数据。
一个角度Winkel α = 0（例如：TRAANG(0)，TRAANG(0,n)）是一个有效的参数设定，并且和旧版本的参数删除不相符。
α允许的值是：
-90度 < α < + 90度



功能

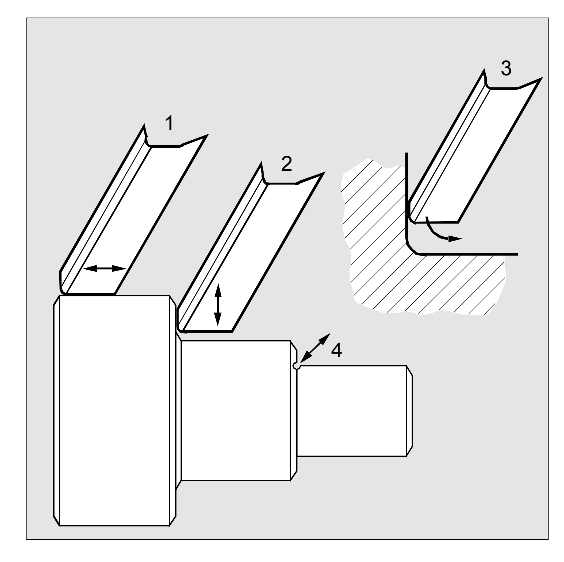
功能斜置轴是为磨削技术而考虑的并且适用于以下操作：

- 用斜置横向进给轴加工
- 对于编程可以使用一个直角坐标系。
- 控制系统将编程设计的直角坐标系的运行转换成实际的加工轴的运行（标准情况）：斜置的横向进给轴。



可以有下列加工：

1. 纵向磨削
2. 平面磨削
3. 磨削一个特定的轮廓
4. 斜置切入磨削



以下设置通过机床数据来确定：

- 在加工轴和斜置轴之间的角度
- 刀具零点位置和在功能“斜置轴”时约定的坐标系原点有关
- 在平行的轴上为平衡运动随时准备好的速度储备。
- 在平行的轴上为平衡运动随时准备好的轴加速度储备。

轴配置

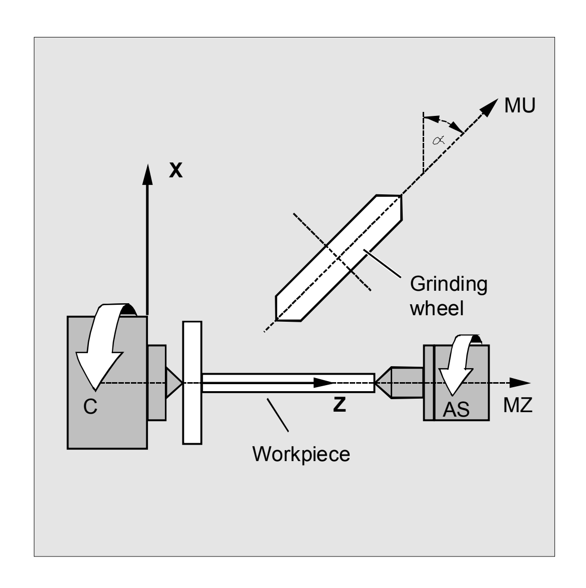
为了可以在直角坐标系中编程，必须通知控制系统坐标系和实际存在的加工轴之间的联系。(MU, MZ)：

- 几何轴的命名
- 几何轴分配给通道轴
  – 普通情况（斜置轴无效）
  – 斜置轴当前有效
- 通道轴分配加工轴编号
- 主轴标记
- 加工轴名称的赋值

操作和标准轴配置的操作相符，“斜置轴有效”例外。



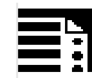
编程举例



N10 G0 G90 Z0 MU=10 G54 F5000 -> -> G18 G64 T1 D1	选择刀具，紧固补偿， 平面选择
N20 TRAANG(45)	斜置轴转换生效
N30 G0 Z10 X5	返回起始位置
N40 WAITP(Z)	释放摆动轴
N50 OSP[Z]=10 OSP2[Z]=5 OST1[Z]=-2 -> -> OST2[Z]=-2 FA[Z]=5000 N60 OS[Z]=1 N70 POS[X]=4.5 FA[X]=50 N80 OS[Z]=0	摆动，直到达到尺寸 （摆动参见第9章）
N90 WAITP(Z)	摆动轴作为定位轴释放
N100 TRAFOOF	关闭转换
N110 G0 Z10 MU=10	空运转
N120 M30	

-> 在一个程序段内编程

7.4.1 编程斜置轴：G05, G07(自软件版本SW 5.3起)



编程

```
G07
G05
```



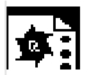
指令说明

G07	返回起始位置
G05	激活斜置切入

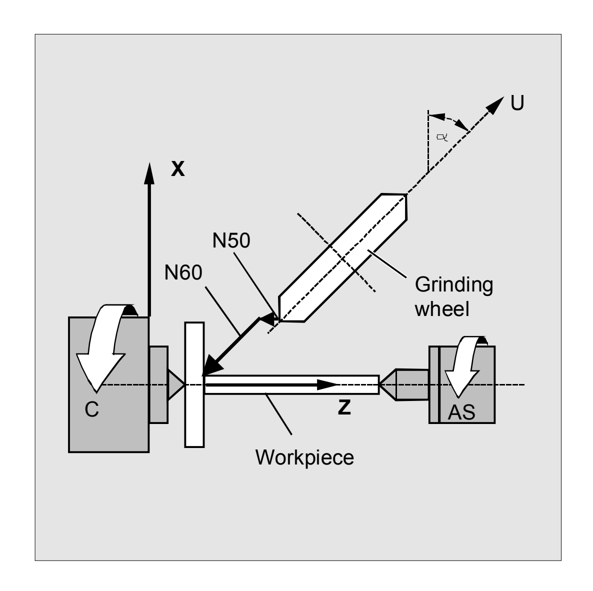


指令G07/G05可以使斜置轴的编程变轻松。
对此可以编程并且显示在直角坐标系中的位置。刀具补偿和零点偏移进行直角计算。NC程序中在斜置轴角度编程之后，可以返回起始位置(G07)并且之后执行斜置切入(G05)。
在手动运行中砂轮可以有选择的直角或者沿斜置轴方向运动（显示的还是直角）。
只有实际的U轴在运动，更新Z轴的显示。

- 重新定位偏移必须以手动方式直角返回。
- 超过直角工作范围的限制以手动方式在当前有效的“PTP运行时”被监控，相应的轴之前被制动。如果“PTP运行”不是当前有效的，那么轴可以准确地一直运行到工作范围极限。
  文献：/FB2/ F2:3－5轴转换，第2章“直角PTP运行”。



编程举例



N..	编程斜置轴角度
N20 G07 X70 Z40 F4000	返回起始位置
N30 G05 X70 F100	斜置切入
N40 ...	

7.5 在选择一个转换时的边界条件

选择转换可以通过零件程序或者MDA。对此要注意

- 不添加一个运行中间程序段（棱角/半径）。
- 必须结束一个样条程序段序列；如果不这样会出现信号提示。
- 刀具精补偿必须取消（FTOCOF）；如果不这样会出现信号提示。
- 刀具半径补偿必须取消（G40）；如果不这样会出现信号提示。
- 一个激活的刀具长度补偿由控制器接收到转换。
- 在转换之前有效的当前框架由控制器取消。
- 一个当前有效的工作范围限制对于和转换有关的轴由控制器取消（和WALIMOF相适应）。
- 取消保护范围监控。

- 轨迹控制运动和精磨被中断。
- 参与转换的轴上的DRF偏移在进刀和主运行加工之间不可以更改（至软件版本SW3）。
- 所有在机床数据中说明的轴必须程序段同步化。
- 将更换的轴换回来；如果不这样，会出现一个信号提示。
- 在不独立的轴时输出一个信号。

换刀

换刀只有在取消刀具半径补偿时才可以。

刀具长度补偿的转换和刀具半径补偿的选择/取消不可以在同一个程序段内编程。

框架转换

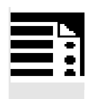
所有只和基准坐标系有关的指令是允许的。一个框架更换在G91（相对尺寸）时不要分开处理，这个与在无效转换时不同。必须运行的增量在新框架的工件坐标系中运用，和前面程序段中什么框架生效没有关系。

排除在外

不可以使用与转换有关的轴

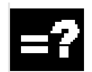
- 作为预设置轴（报警）
- 针对返回固定点（报警）
- 用于返回基准点（报警）

7.6 取消转换： TRAFOOF



编程

TRAFOOF



指令说明

TRAFOOF	关闭所有当前有效的转换/框架

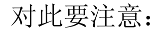


功能

用指令TRAFOOF使所有当前有效的转换和框架关闭。



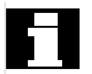
之后所需的框架必须通过更新的编程生效。



对此要注意：

对于取消转换的限制和选择转换时的限制相同（参见前面的“在选择转换时的边界条件”一章）

7.7 级联的转换

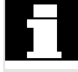


自软件版本SW5起可以每**两个**转换连续转换（级联的），以至于第一转换中轴运行的分量是级联的第二程序段的输入数据。第二转换的运行分量影响加工轴。

- 级联在软件版本SW5中可以包括**两个**转换。
- 第二转换必须是“**斜置轴**”（TRAANG）。
- 作为第一个转换可以：
  - 定向转换TRAORI（TRAORI），
    包括万向铣削头
  - TRANSMIT
  - TRACYL
  - TRAANG

应用

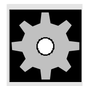
-轮廓的磨削，轮廓作为一个圆柱展开的外形轮廓线用一个斜置的砂轮，例如：刀具磨削，来编程。
-用斜置砂轮精加工一个不圆轮廓，这个不圆轮廓是用TRANSMIT生成的。



使用一个级联转换的开通指令的前提是，单独的必须级联的转换和必须激活的级联的转换通过机床数据来定义。
转换详细描述中说明的边界条件和特殊情况在使用级联时也需要注意。

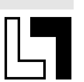


其它说明
关于转换机床数据的设计可以在功能描述中找到：M1和F2。



机床制造商(MH7.1)

请注意机床制造商的说明，有可能通过机床数据在前面定义的转换。



转换和级联的转换是选项。有关特定控制系统中级联转换的可用性，分别在各个目录中给出信息。

对于级联的转换指令：

TRACON用于开通

TRAFOOF用于关闭。

开通



编程

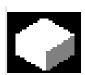
`TRACON(trf, par)`	一个级联的转换开通。



参数说明

`trf`	级联转换的编号： 0或者1对于第一/唯一的级联的转换。 如果在这个位置上没有编程，那么值的说明0或者1的意思是相同的，也就是说激活第一个/唯一的转换。 2对于第二个级联的转换。 （值不等于0－2会发出一个错误报警）。

par	一个或者几个通过逗号分隔的在级联中的转换参数，例如：斜置轴的角度。在没有设置参数时，预设置或者最后使用的参数有效。如果前面的参数预置应该有效，那么设置逗号时必须考虑给出的参数要按顺序求值，参数是以这个顺序排序的。特别是在说明至少一个参数时在参数前必须有逗号，即使trf的说明不是必须的，例如：TRACON(, 3.7)。



功能

一个级联的转换开通。另一个之前有效的转换通过TRACON()隐含关闭。

一个刀具总是分配到级联的第一个转换。后来的转换会这样运行，就好像当前有效的刀具长度是零。只有通过机床数据设置的刀具(_BASE_TOOL_)的基准长度对于级联的第一个转换有效。

关闭



编程

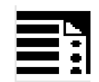
```
TRAFOOF
```



功能

这一指令使最后开通的（级联的）转换关闭。

7.8 可转换的几何轴，GEOAX



编程

```
GEOAX(n,通道轴,n,通道轴,…)

GEOAX()
```



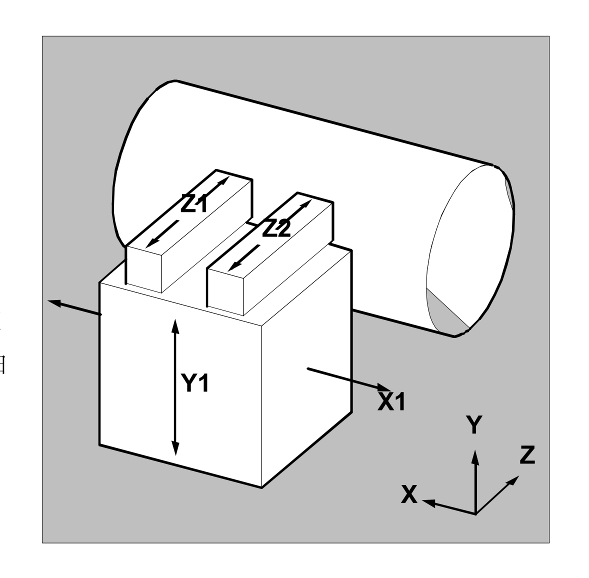
参数说明

GEOAX (n,Kanalachse,n,Kanalachse,…)	几何轴转换。
GEOAX()	调入几何轴基本配置
n	几何轴编号（n=1，2或者3），另一个通道轴应分配到这个几何轴。 n=0：从几何轴组合中没有替换地清除说明的通道轴。
通道轴	通道轴的名称，这个通道轴应被纳入几何轴组合中去。



功能

用功能“可转换的几何轴”可以从零件程序出发更改通过机床数据配置的几何轴组合。对此一个定义为同步附加轴的通道轴可以替换任意一个几何轴。



举例：

刀具溜板可以通过通道轴X1，Y1，Z1，Z2来运行。在零件程序中可以使用轴Z1和Z2交替地作为几何轴Z。轴之间的转换在零件程序中用GEOAX进行。

开通之后X1，Y1，Z1的组合有效（通过MD可设置）。

N100 GEOAX (3,Z2)	通道轴Z2作为Z轴起作用
N110 G1	
N120 GEOAX (3,Z1)	通道轴Z1作为Z轴起作用



工作流程

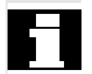
几何轴编号

在指令GEOAX（n，通道轴...）中编号n表示几何轴，后面说明的通道轴从属于这个几何轴。

对于一个通道轴的转换几何轴编号1至3（X-, Y-, Z-轴）是允许的。

用n=0从几何轴组合中清除一个没有用几何轴重新覆盖的赋值的通道轴。

一个由几何轴组合中的转换替代的轴在转换过程之后，通过它们通道轴名称作为附加轴可编程。



所有的框架，保护范围和工作范围限制都可以用几何轴转换来删除。

极坐标：

几何轴的交换用GEOAX模拟一个平面交换(G17-G19)将模态的极坐标设置为值0。

DRF, NPV:

一个可能发生的手轮偏移（DRF）或者一个外部的零点偏移在转换之后依旧有效。

交换轴位置

通过对已经赋值的通道轴的轴编号的新的指示可以在几何轴组合中进行一个位置转换。

N... GEOAX (1, XX, 2, YY, 3, ZZ)	通道轴XX是第一，YY是第二，ZZ是第三几何轴，
N... GEOAX (1, U, 2, V, 3, W)	通道轴U是第一，V是第二，W是第三几何轴。



前提条件和限制

1. 几何轴的切换在下列情况时不适用：
   - 有效的转换，
   - 有效的样条插补，
   - 有效的刀具半径补偿，
   - 有效的刀具精补偿 （编程说明基础部分：第8章）
2. 如果几何轴和通道轴显示相同的名称，那么不可以转换每个几何轴。 （编程说明基础部分：第8章）。
3. 参与转换的轴都不可以参与作用，此作用可以超越程序段限制持续下去，例如：使用类型A的定位轴或者使用跟随轴时可以。
4. 用指令GEOAX只能在已经生效时替换现有几何轴（不需补充定义新的几何轴）。
5. 用GEOAX在处理**轮廓表格**期间（CONTPRON，CONTDCON），交换轴会导致报警。

使转换无效

指令GEOAX()调入几何轴组合的基本配置。
在POWERON之后并且在转换到参考点运行方式时将自动转换回基本配置。



其他说明

转换过程和刀具长度补偿

一个当前有效的刀具长度补偿在转换过程之后也是有效的。尽管如此，它对新接纳或者交换位置的几何轴仍然有效，当它们还没有运行时。使用第一个针对这些几何轴的运行指令时，生成的运动行程相应的由刀具长度补偿的总和与编程设计的运动行程组成。
在转换时在轴组合中保持自身位置的几何轴，也保持其状态，包括刀具长度补偿。

几何轴配置和转换切换

在一个有效的转换中所适用的几何轴配置（通过机床数据确定），不可以通过功能“可转换的几何轴”来更改。

如果需要改变和转换相关联的几何轴配置，那么这只有通过其它的转换才可以。

可以用激活转换来删除通过GEOAX改变的几何轴配置。

如果所设置的机床数据对于转换和对于几何轴的转换相互矛盾，那么在转换中的设置有优先权。

举例：

一个转换有效。根据机床数据转换在复位时保持不变，同时在复位时还生成几何轴的基本配置。在这种情况下几何轴配置和其随转换而确定的配置一样保持不变。





编程举例

一个机床有6个通道轴，名称是XX，YY，ZZ，U，V，
W。通过机床数据的几何轴配置的基本设置是：
通道轴XX = 第1几何轴（X轴）
通道轴YY = 第2几何轴（Y轴）
通道轴ZZ = 第3几何轴（Z轴）

N10	GEOAX()	几何轴的基本配置有效。
N20	G0 X0 Y0 Z0 U0 V0 W0	所有轴快速运行到位置0。
N30	GEOAX(1,U,2,V,3,W)	通道轴U成为第一（X），V成为第二（Y），W成为第三几何轴（Z）。
N40	GEOAX(1,XX,3,ZZ)	通道轴XX成为第一（X），ZZ成为第三几何轴（Z）。通道轴V保持第二几何轴（Y）不变。
N50	G17 G2 X20 I10 F1000	在X，Y平面上的整圆。运行通道轴XX和V
N60	GEOAX(2,W)	通道轴W成为第二几何轴（Y）。
N80	G17 G2 X20 I10 F1000	在X，Y平面上的整圆。运行通道轴XX和W。
N90	GEOAX()	复位到基本状态
N100	GEOAX(1,U,2,V,3,W)	通道轴U成为第一（X），V成为第二（Y），W成为第三几何轴（Z）。
N110	G1 X10 Y10 Z10 XX=25	通道轴U，V，W分别运动到位置10，XX作为附加轴运动到位置25。
N120	GEOAX(0,V)	从几何轴组合中取出V。此外U和W是第一（X）和第三几何轴（Z）。第二几何轴（Y）保持未覆盖。
N130	GEOAX(1,U,2,V,3,W)	通道轴U保持第一（X），V成为第二（Y），W保持第三几何轴（Z）。
N140	GEOAX(3,V)	V成为第三几何轴（Z），W被修改并且从几何轴组合中取出。第二几何轴（Y）和以前一样未覆盖。

■

用于记录

刀具补偿

8.1 补偿存储器

建立补偿存储器

每个数据范围都可以用T和D编号来调入（除了“平面的D编号”）并且除了刀具的几何尺寸说明之外还包含其他的输入，例如：刀具类型。

自软件版本SW 4起

如果刀具管理在NCK之外进行，那么使用“面积D编号结构”。在这种情况下带从属刀具补偿程序段的D编号不对刀具进行赋值。
在零件程序中可以进一步编程T。但是这个T和编程设计的D编号没有关系。

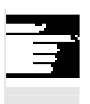
对于几何尺寸大小（例如：长度1或者半径）存在几个输入分量。这些被加性的计算成一个生成的有效的大小（例如：总长度1，总半径）。

不需要的补偿可以用值零来覆盖。



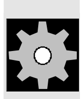
补偿存储器P1到P25的单独的值通过程序的系统变量可读并且可写。

刀具参数 编号(DP)	系统变量的意义	附注
$TC_DP 1	刀具类型	概要参见清单
$TC_DP 2	切削刃位置	只针对车刀
几何尺寸	**长度补偿**	
$TC_DP 3	长度1	根据类型和平面的计算
$TC_DP 4	长度2	
$TC_DP 5	长度3	
几何尺寸	**半径**	
$TC_DP 6	半径	
$TC_DP 7	切槽锯片的槽宽b，铣削刀具的倒圆半径	
$TC_DP 8	超出规定范围k	只针对切槽锯片
$TC_DP 11	圆锥形铣削刀具角度	
磨损	**长度和半径补偿**	
$TC_DP 12	长度1	
$TC_DP 13	长度2	
$TC_DP 14	长度3	
$TC_DP 15	半径	
$TC_DP 16	切槽锯片的槽宽b，铣削刀具的倒圆半径	
$TC_DP 17	超出规定范围k	只针对切槽锯片
$TC_DP 20	圆锥形铣削刀具角度	
基础尺寸/适配器	**长度补偿**	
$TC_DP 21	长度1	
$TC_DP 22	长度2	
$TC_DP 23	长度3	
工艺		
$TC_DP 24	后角	针对车刀



其它说明

所有其他的参数被保留。



机床制造商

通过MD可以配置使用者切削数据。

8.2 刀具管理的语言指令



指令说明

T="WZ"	带名称的刀具的选择
NEWT ("WZ",DUPLO_NR)	设置新的刀具，可选择双编号
DELT ("WZ",DUPLO_NR)	删除刀具，可选择双编号
GETT ("WZ",DUPLO_NR)	决定T编号
SETPIECE (x,y)	设置件数
GETSELT (x)	读取前面选出的刀具编号（T编号）
"WZ"	刀具标记
DUPLO_NR	件数
x	主轴编号，可选择说明



在使用刀具管理时，可以设置刀具名称并且调入，
例如：T="BOHRER"或者T="123"。



NEWT-功能

用NEWT功能可以在NC程序段中设置一个新刀具的名称。
功能作为返回参数自动提供了生成的T编号，用这个编号
可以接下来给刀具定地址。

返回参数=NEWT("WZ", DUPLO_NR)

如果没有第二编号的说明，那么这个在刀具管理中生
成。

举例：

DEF INT DUPLO_NR	
DEF INT T_NR	
DUPLO_NR = 7	
T_NR=NEWT("BOHRER", DUPLO_NR)	设置带第二编号7的新刀具“BOHRER”。 生成的T编号存储在T_NR中。

DELT-功能

用DELT功能可以不考虑T编号删除一个刀具。

DELT("WZ", DUPLO_NR)

GETT-功能

GETT功能给一个刀具提供了设置刀具数据必需的T编号，这个刀具通过名称是已知的。

返回参数=GETT("WZ", DUPLO_NR)

如果有给出名称的几个刀具，那么送回最有可能的刀具的T编号。

返回 = –1:可以不给刀具分配刀具名称或者第二编号。

举例：

T="BOHRER"	
R10=GETT("BOHRER", DUPLO_NR)	求得带第二编号=DUPLO_NR的BOHRER的T编号

在此之前必须用NEWT或者$TC_TP1[]使"BOHRER"已知。

$TC_DP1[GETT("BOHRER", DUPLO_NR),1]=100	写入带刀具名称的刀具参数（系统变量）

SETPIECE-功能

这个功能用于件数监控数据的更新。
功能包含所有的刀具切削，切削从最后激活的SETPIECE转换到命名的主轴编号。

SETPIECE(x,y)

x	完成的工件数量
y	y 主轴编号，0代表主主轴（标准设置）

GETSELT-功能

功能提供了为主轴选出的刀具的T编号。
因此在M6之前就可以存取刀具的补偿数据并且可以早一些生成和主运行的同步。

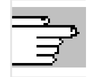
用刀具管理换刀的举例

T1 刀具预选；也就是说，这个刀库可以并行的用于在刀具位置上的加工。

M6 一个预选刀具的转换（分别根据在机床数据中的预设置也可以在没有M6的情况下来编程）。

举例：

`T1 M6`	换刀具1
`D1`	选择刀具长度补偿
`G1 X10 …`	用T1工作
`T="BOHRER"`	刀具预选 钻头
`D2 Y20 …`	刀沿转换T1
`X10 …`	用T1工作
`M6`	换入刀具钻头
`SETPIECE(4)`	完成的工件数量
`D1 G1 X10 …`	用钻头工作



刀具管理所有变量完整的清单可以在附录中的系统变量清单中找到。

8.3 在线-刀具补偿PUTFTOCF, PUTFTOC, FTOCON, FTOCOF



编程

```
FCTDEF(多项式编号, LLimit, ULimit,a0,a1,a2,a3)
PUTFTOCF (多项式编号, 参考值, 长度1_2_3, 通道, 主轴)
PUTFTOC (值, 长度1_2_3, 通道, 主轴)
FTOCON
FTOCOF
```

指令说明

PUTFTOCF	连续写入在线－刀具补偿
FCTDEF	功能PUTFTOCF的参数化
PUTFTOC	不连续写入在线－刀具补偿
FTOCON	在线－刀具补偿开通
FTOCOF	在线－刀具补偿关闭

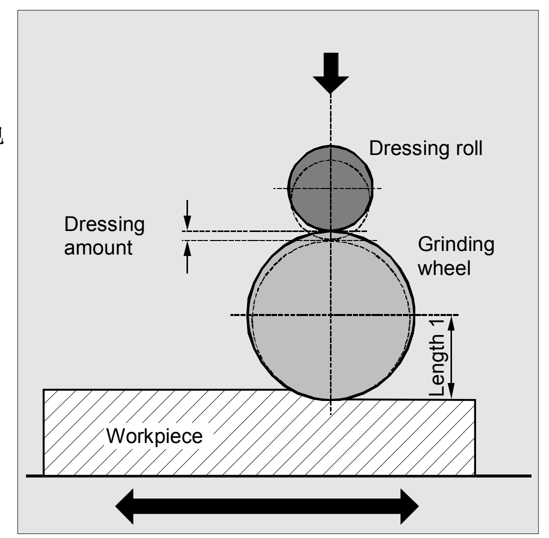
参数说明

多项式编号	值1至3：同时可以是最多3级多项式；多项式至第三级
参考值	参考值，从中可以导出补偿值
长度1_2_3	磨损参数，在其中添加刀具补偿值
通道	通道的编号，在通道中刀具补偿有效；只有当不止涉及自身通道时才给出说明
主轴	主轴的编号，对于主轴在线刀具补偿有效；在没有当前有效的砂轮的情况下需要给出说明
LLimit	上限值
ULimit	下限值
a_0,a_1,a_2,a_3	多项式功能的系数
值	在磨损参数中添加的值



功能

用这个功能可以通过一个在线有效的刀具长度补偿立刻计算出由加工得出的刀具补偿（例如：CD修整：并行地用于加工来修整砂轮）。刀具长度补偿可以从加工通道或者一个平行的通道中（修整通道）来改变。





在线－WZK只能用在磨具上。

在线WZK概论

分别根据修整进程的时间点，对于在线WZK的写入使用不同的功能：

- 程序段方式连续写入：PUTFTOCF
- 模态方式连续写入：ID=1 DO FTOC
  (参见同步动作章节)
- 不连续写入： PUTFTOC

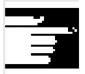
在使用连续写入时（每个IPO-节拍），为了避免额定值的跳跃，在计算功能开通之后每个改变在磨损存储器中是加法计算的。
以下总是有效的：
在线WZK可以在每个通道中对于每个主轴和磨损参数的长度1, 2或者3有效。

几何轴长度的赋值根据当前平面来进行。

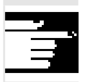
如果和当前有效的砂轮没有关系的话，到刀具的主轴分配通过刀具数据在GWPSON或者TMON时进行（参见编程指南“基础”）。
当前砂轮面或者左侧砂轮面的磨损参数总是在没有激活的刀具时补偿。



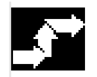
在几个砂轮面的同一补偿时，通过级联规定（描述参见操作）必须设法自动保存第二个砂轮面的值。



如果对于一个加工通道规定在线补偿，那么可以在这个通道中不由加工程序或者通过操作来改变当前刀具的磨损值。



对于恒定砂轮圆周速度(SUG)以及刀具监控TMON和无心磨削CLGON要考虑到在线WZK。



工作流程

PUTFTOCF = 连续写入

修整进程和加工同时进行：

通过砂轮总宽度用修整轮或者用修整金刚钻从砂轮的一面向另一面修整。

加工和修整可以在不同的通道中进行。如果没有编程通道，那么补偿在当前有效的通道中有效。

```
PUTFTOCF(多项式编号, 参考值, 长度1_2_3, 通道, 主轴)
```

刀具补偿在加工通道中连续地根据第1，第2，或者第3级的多项式功能来更改，这个功能之前必须用FCTDEF来定义。

补偿由变量“参考值”导出，例如：变化的实际值。

如果没有编程主轴编号，那么修正当前有效的、使用中的刀具。

确定功能FCTDEF的参数

需要在一个自身的程序段中给定参数。

FCTDEF (多项式编号, LLimit, ULimit, a_0,a_1,a_2,a_3)

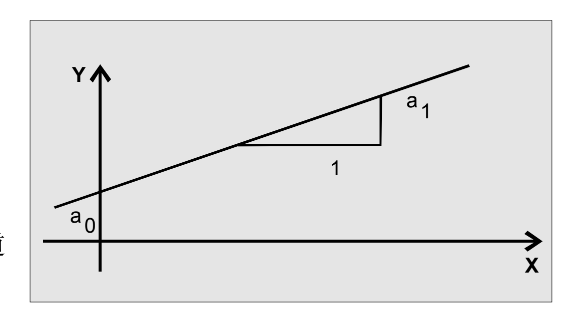
多项式可以是第1，第2或者第3级。

极限表示极限值（LLimit = 下限，ULimit = 上限）

举例：

带导程1的直线(y = a0 + a1x)

```
FCTDEF(1, -1000, 1000, -$AA_IW[X], 1)
```



在线WZK不连续写入：PUTFTOC

用这个指令可以一次编写一个补偿值。补偿在目标通道中立即有效。

PUTFTOC的应用：

砂轮从一个并行的通道中来修整，尽管如此它和加工是不同时进行的。

```
PUTFTOC(值, 长度1_2_3, 通道, 主轴)
```

规定长度1，2或者3的在线WZK改变规定的值，也就是说会在磨损参数中添加这个值。

计算在线-WZK： FTOCON, FTOCOF

目标通道只在FTOCON有效的时候才能接受在线刀具补偿。

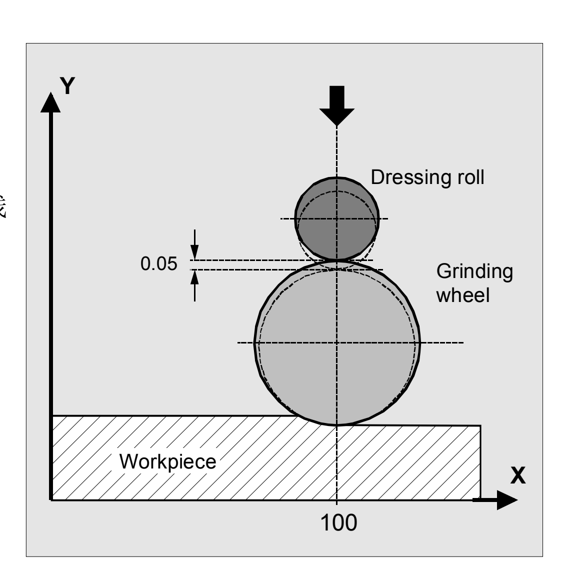


- FTOCON必须在通道中编写，在这个通道中补偿应该有效。
  用FTOCOF不能继续得出补偿值，尽管如此在切削专用的补偿数据中全部用PUTFTOC编写的值被修正。
- FTOCOF总是复位位置。
- PUTFTOCF总是程序段方式有效，也就是说，在相邻的运行程序段中。
- 用FTOC也可以模态地选择在线刀具补偿。对此更多的信息参见“运动同步动作”章节。

编程举例

任务

在使用带有后来确定的平面磨削机床时，在磨削运动开始之后X100时将砂轮的值修整0.05。修整量应该用在线WZK来编写，在使用磨削工具时连续有效。



Y:砂轮的横向进给轴

V:修整轮的横向进给轴

加工：通道1用轴X，Z，Y

修整：通道2用轴V

在通道1中的加工程序：

%_N_BEARB_MPF	
…	
N110 G1 G18 F10 G90	基本设定
N120 T1 D1	选择当前刀具
N130 S100 M3 X100	主轴开，运行到起始位置
N140 INIT (2, "ABRICHT", "S")	在通道2中选择修整程序
N150 START (2)	在通道2中起动修整程序
N160 X200	运行到目标位置
N170 FTOCON	在线补偿开通
N… G1 X100	再加工
N…M30	

在通道2中的修整程序：

%_N_ABRICHT_MPF	
…	
N40 FCTDEF (1, -1000, 1000, -$AA_IW[V], 1)	定义功能：直线
N50 PUTFTOCF (1, $AA_IW[V], 3, 1)	连续写入在线－WZK： 由V轴运动导出，当前砂轮的长度3在通道1中修正。
N60 V-0.05 G1 F0.01 G91	用于修整的横向进给运动，PUTFTOCF只在这个程序段内有效
…	
N… M30	

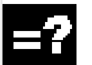
修整程序模态有效：

%_N_ABRICHT_MPF	
FCTDEF(1,-1000,1000,-$AA_IW[V],1)	定义功能。
ID=1 DO **FTOC(**1,$AA_IW[V],3,1**)**	选择在线刀具补偿： V轴的实际值是多项式1的输入值；结果在通道1中作为补偿值添加到当前有效的砂轮长度3中。
WAITM(1,1,2)	和加工通道同步
G1 V-0.05 F0.01, G91	修整的横向进给运动
G1 V-0.05 F0.02	
...	
CANCEL(1)	取消在线－补偿
...	

8.4 刀具半径补偿保持恒定，CUTCONON（自软件版本SW 4起）

编程

```
CUTCONON
CUTCONOF
```



说明

CUTCONON	功能刀具半径补偿开通保持恒定
CUTCONOF	功能刀具半径补偿关闭保持恒定（标准设置）



功能

功能“刀具半径补偿保持恒定”用于，抑制程序段数目的刀具半径补偿，然而在补偿时，在编程设计的和刀具中心点实际开始运动的轨迹之间的分差作为偏移保持，这个分差是通过刀具半径补偿在前面的程序段中生成的。

如果在换向点行铣削时需要几个运行程序段，而这些程序段不是刀具半径补偿生成的轮廓（绕行策略）所需要的程序段，那么可以设置这个功能，是有利的。

它的设置不取决于刀具半径补偿的类型
（$2^1/_2$D, 3D-端铣，3D-圆周铣削）



工作流程

在通常情况下在激活补偿抑制之前，刀具半径补偿已经是有效的，并且如果取消补偿抑制的话依旧有效。

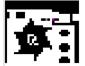
在最后运行程序段在CUTCONON之前，会运行到程序段终点的偏移点。

所有后续的，并且在其中补偿抑制当前有效的程序段可以在没有补偿的情况下运行。

然而它们在运行时会从最后补偿程序段的终点偏移一定的矢量到它的偏移点。

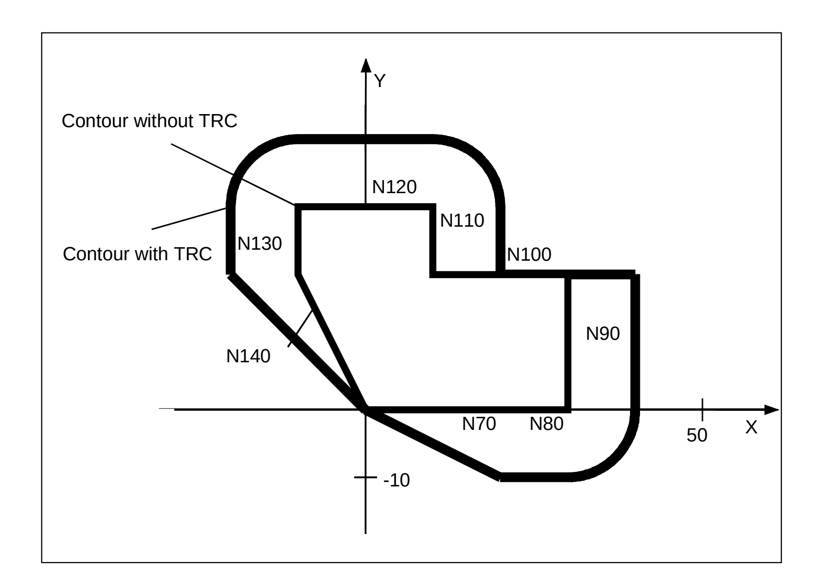
这些程序段的插补类型（线性的，圆周的，多项式的）时任意的。

补偿抑制的取消程序段，也就是说包含CUTCONOF的程序段，进行标准修正；它在起始点的偏移点开始。
在过程程序段的终点，也就是说最后编程设计的带有有效CUTCONON的运行程序段的终点，和这个点之间插入一个线性程序段。
圆弧程序段，在这些程序段时圆弧平面垂直于补偿平面（垂直的圆弧），在操作这些程序段时，就好像在其中编程了CUTCONON。
补偿抑制的隐含的激活在第一运行程序段中自动清除，这个运行程序段在补偿平面中包含一个运行程序段并且不是这样的圆弧。
在这个意义上垂直的圆弧只可能在圆周铣削时出现。



举例

N10 ;	刀具d1的定义
N20 $TC_DP1[1,1]= 110 ;	类型
N30 $TC_DP6[1,1]= 10. ;	半径
N40	
N50 X0 Y0 Z0 G1 G17 T1 D1 F10000	
N60	
N70 X20 G42 NORM	
N80 X30	
N90 Y20	
N100 X10 CUTCONON;	补偿抑制开通
N110 Y30 KONT ;	在补偿抑制关闭时如果有可能插入绕行圆弧
N120 X-10 CUTCONOF	
N130 Y20 NORM ;	没有绕行圆弧或者WRK的关闭
N140 X0 Y0 G40	
N150 M30	





其它说明

1. 如果没有刀具半径补偿是有效的（G40），
   那么CUTCONON无效。不产生报警。
   但G代码保持有效。如果在一个后来的程序段中用
   G41或者用G42来使刀具半径补偿开通，那么这是有
   意义的。
2. 在第7G代码组（刀具半径补偿；G40/G41/G42）的
   G代码的转换在当前有效的CUTCONON时是允许的。
   G40之后转换立即有效。
   在此得出偏移，过程程序段以这个偏移来运行。
3. 如果CUTCONON或者CUTCONOF在一个不带运行
   的程序段中在当前有效的补偿平面内编程，那么有效
   性延迟到下一个带有这样运行的程序段。

更多信息：/FB/, W1 刀具补偿

8.5 激活3D-刀具补偿CUT3DC, CUT3DF, CUT3DFS/CUT3DFF

说明

CUT3DC	激活圆周切削的3D半径补偿
CUT3DFS	对于带恒定方向的端铣的3D刀具补偿刀具定向通过 G17 - G19来确定并且不受框架的影响。
CUT3DFF	对于带恒定方向的端铣的3D刀具补偿刀具定向通过 G17 - G19来确定并且如果有可能是通过框架旋转的方向。
CUT3DF	对于带定向改变的端铣的3D刀具补偿（只在有效的5轴转换时）。
CUT3DCC	对于带限制面积的圆周铣削的3D刀具补偿.
CUT3DCCD	对于带限制面积的用分差刀具的圆周铣削的3D刀具补偿。
G40 X Y Z	用于关闭：带几何轴的线性程序段G0/G1
ISD=Wert	浸没深度



指令模态有效并且在同一组，如同CUT2D和CUT2DF。

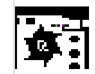
在当前平面随着下一次运动才可以取消。这个总是适用于G40并且不取决于CUT指令。



功能

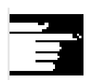
在圆柱型刀具的刀具半径补偿时，要考虑可变的刀具方向。

对于3D刀具半径补偿的选择适用和在2D刀具半径补偿时一样的程序指令。用G41/G42说明在左/右运动方向的补偿。返回特性总是NORM。



举例

N10 A0 B0 X0 Y0 Z0 F5000	
N20 T1 D1	刀具调入，调入刀具补偿值
N30 TRAORI(1)	转换选择
N40 CUT3DC	3D-刀具半径补偿-选择
N50 G42 X10 Y10	刀具半径补偿-选择
N60 X60	
N70 …	



其它说明

中间程序段在有效的3D刀具半径补偿时是允许的。
2 1/2D-刀具半径补偿的确定有效。

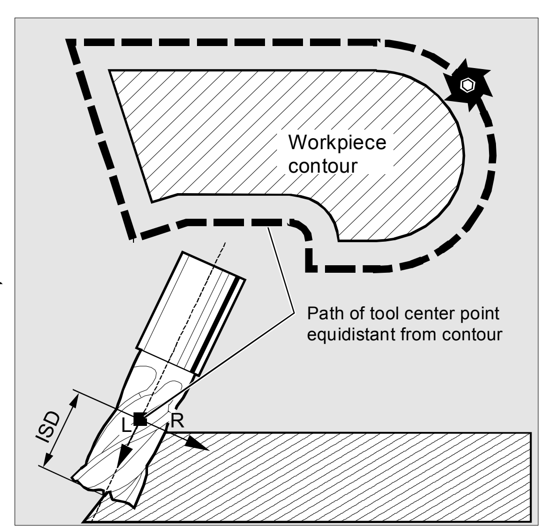
3D刀具半径补偿只在选择的5轴转换时有效。

在外角处总是插入一个圆弧程序段。G450/G451没有意义。

没有运用指令DISC。

2 1/2D-和3D-刀具半径补偿之间的区别
在3D刀具半径补偿时刀具方向是可以更改的。

在2 1/2D刀具半径补偿时只计算一个刀具的恒定方向。



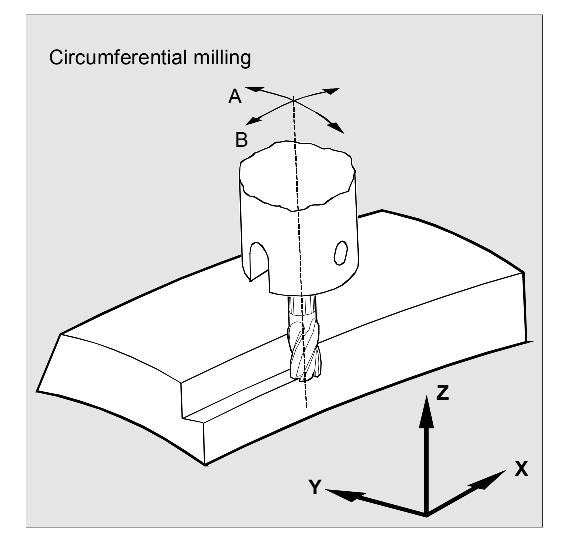


3D刀具半径补偿也可以作为5D补偿，因为在这种情况下在空间中刀具位置的5个自由度可供支配。

8.5.1 3D-刀具半径补偿:圆周铣削，端铣

圆周铣削

这里使用的圆周铣削的变量通过一条轨迹和其方向的说明来实现。在这种加工类型时，在轨迹上刀具类型没有意义。关键只是刀具啮合点处的半径。





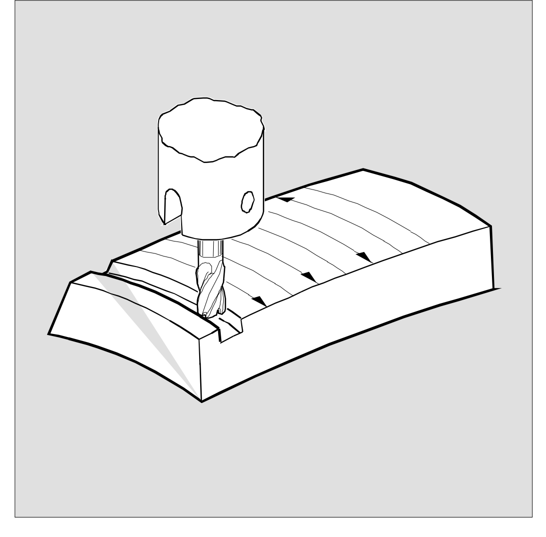
功能3D-WRK限制使用在圆柱形刀具上。

端铣

对于这种3D铣削类型需要在工件表面积上按行描述3D轨迹。

计算需要考虑到刀具类型和刀具尺寸通常在CAM中执行。

后置处理程序将刀具定向（在当前有效的5轴转换时）和所需的3D刀具补偿的G代码写入零件程序，除了NC程序段。



因此机床操作员有可能使用比较小的刀具，偏离用于NC轨迹计算的刀具。



举例：

用铣刀10毫米计算NC程序段。

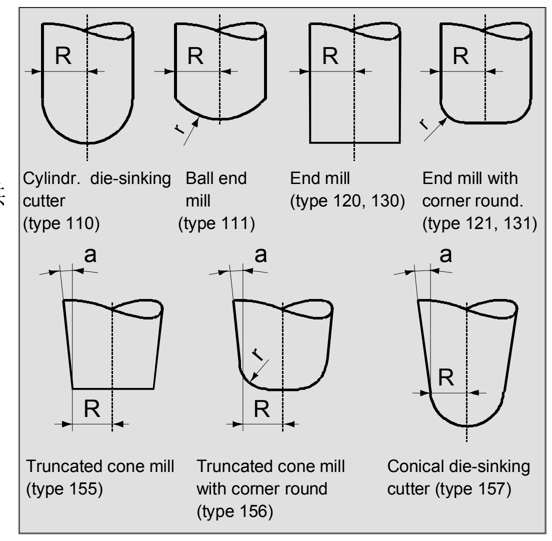
这里也可以用直径9.9毫米的铣刀来完成，对此可以用变化了的成型材料的粗糙度来计算。

8.5.2 刀具类型/带变化了的尺寸的刀具转换G40/41/42

铣刀类型，刀具数据

在这张表格里汇集了对于端铣有可能的刀具类型和刀具数据的极限值。

不考虑刀柄的类型，刀具类型120和155作用是相同的。如果在NC程序中说明了另一个类型编号，和表格中提供的编号不同，系统自动使用圆柱形模具铣刀的刀具类型110。超过刀具数据的极限值时产生报警。



铣刀类型	类型号	R	r	a
圆柱形模具铣刀	110	>0	X	X
圆锥头铣刀	111	>0	>R	X
带柄铣刀，角度铣刀	120, 130	>0	X	X
带柄铣刀，带刀尖圆弧的角度铣刀	121, 131	>r	>0	X
截锥铣刀	155	>0	X	>0

刀具数据	刀具参数	
刀具尺寸	几何尺寸	磨损
R	$TC_DP6	$TC_DP15
r	$TC_DP7	$TC_DP16
a	$TC_DP11	$TC_DP20

X = 未计算

R = 刀柄半径（刀具半径）

r = 角半径

a = 刀具纵向轴和环面上面终端之间的角度

刀具长度补偿

作为长度补偿的参考点，刀尖有效（交点纵向轴/表面积）。

3D-刀具补偿，换刀

一个带有变化了的尺寸（R，r，a）的新刀具或者另一种类型的新刀具只能用G41或者G42的编程来说明（G40到G41或者G42的过渡更新G41或者G42的编程）。

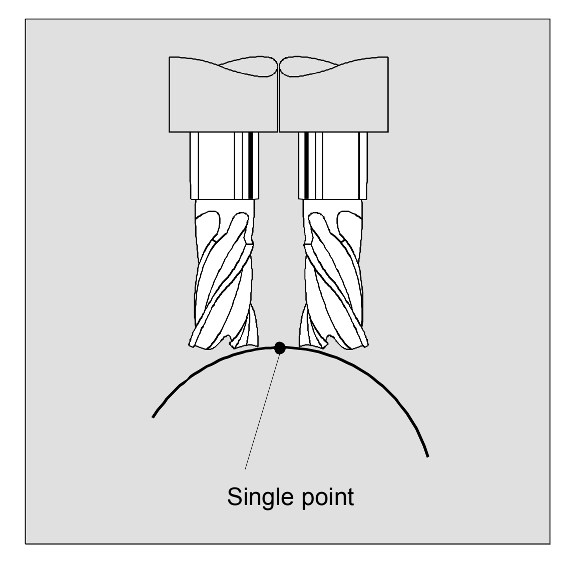
所有其他的刀具数据，例如：刀具长度，在这种情况下依旧不考虑，以至于这样的刀具在没有更新的G41或者G42的情况下也可以转换。

8.5.3 轨迹上的补偿，轨迹曲线和浸没深度ISD

轨迹上的补偿

在端铣时必须注意接触点在刀具表面积上跳跃的情况。就像在这个例子里用垂直的刀具在进行凸起面积的加工时。

图中所示的应用可以视为极限情况。

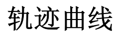


这个极限情况由控制器监控，通过在角度定位的基础上识别在刀具和面积标准矢量之间跳跃式的加工点的变化。

在这些位置上控制器添加线性程序段，以至可以执行运动。

对于计算线性程序段，允许的角度范围储存在侧向角的机床数据中。

如果超过在机床数据中确定的允许角度范围的极限值，那么系统会报警。



轨迹曲线

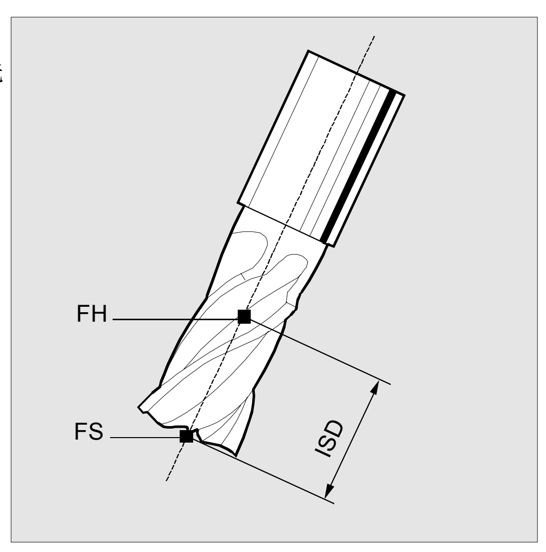
不监控轨迹曲线。这里只适用这样的刀具，用这样的刀具工作可以没有轮廓失真。

浸没深度ISD

用程序指令ISD（插入深度）给刀具的浸没深度在圆周铣削时编程。因此由可能在刀具的表面积上改变加工点的位置。

ISD规定在铣刀刀尖(FS)和铣刀辅助点(FH)之间的距离。点FH通过编程设计的加工点在刀具轴上的投影产生。

ISD只有在当前有效的3D刀具半径补偿时运用。

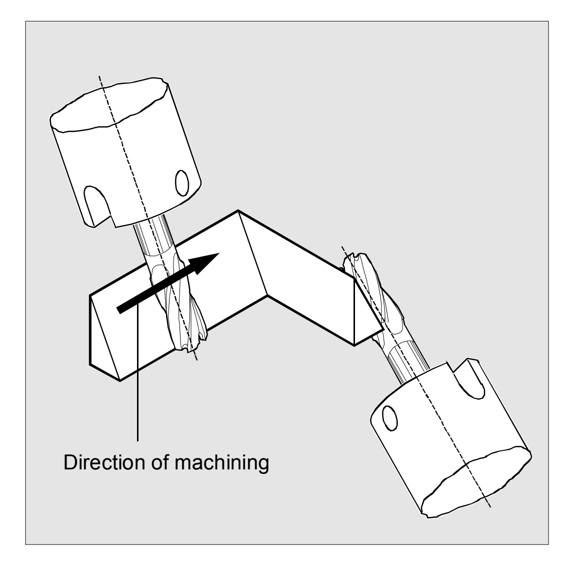


8.5.4 内角/外角和交点运行，G450/G451（自软件版本SW 5起)

内角/外角

外角和内角分开操作。内角或者外角的名称取决于刀具方向。

在一个角处的定向改变时可能出现这样的情况，角类型在加工期间自己改变。如果出现这样的情况，那么产生错误报告并且加工中断。



3D补偿的交点运行：

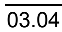
在3D圆周铣削时，现在在外角处运用G代码G450/G451，也就是说，可以返回偏移曲线的交点。至软件版本SW4总是在外角处插入一个圆弧。

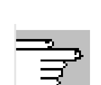
自软件版本SW5起可以使用的交点运行，在典型的CAD生成的3D程序时特别有利。这些经常由短的直线程序段组成（用于平滑曲线的近似值），在这些程序段时相邻程序段之间的过渡几乎是切线的。

在轮廓外面进行刀具半径补偿时，插入用于外角绕行的迄今原则上的圆弧。因为这些程序段在几乎切线的过渡时非常短，产生不受欢迎的速度干扰。

在这些情况下模拟2 ½ D半径补偿两个参与的曲线延长，并且返回两条延长曲线的交点。

通过延长两个参与程序段的偏移曲线，并且在垂直于角上刀具方向的平面上确定曲线的交点，以此来决定交点。如果没有这样的交点，那么角就象以前那样操作，也就是说不插入圆弧。





交点运行的更多的信息

/FB/ W5, 3D-刀具半径补偿

8.5.5 带有限制面积的3D圆周铣削，CUT3DCC，CUT3DCCD



功能

由CAD系统产生的NC程序通常情况下用大量较短线性程序段近似于一个标准刀具的中心点轨迹。为了使这些生成的很多零件轮廓的程序段尽可能准确地模拟原始轮廓，必需在零件程序中进行某个匹配。

对于最优补偿所需的，但是在零件程序种不可用的重要信息必须用合适的措施来替代。后面表示出独特的方法，为了平衡临界的过渡，要么

- 直接在零件程序中，或者
- 借助实际轮廓，例如：通过刀具的横向进给。

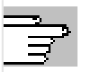
除了对于独特使用示例的操作也会遇到以下情况的操作，在那些情况时标准刀具不是描述中心点轨迹，而是描述在加工面上的轮廓。在这种情况下限制面积和刀具无关。和在习惯的刀具半径补偿时一样，总半径用于计算限制面积上的垂直偏移的计算。

用标准刀具的3D半径补偿：

在带有连续或者恒定的刀具定向改变的3D圆周铣削时，经常编程一个定义的标准刀具的刀具中心点轨迹。因为实际操作上合适的标准刀具常常不可以使用，可以使用一个和标准刀具偏差不是太多的刀具。

带斜槽壁的槽铣削，用于用CUT3DC的圆周铣削

通过在必须加工的面积的面积标准的方向上进行横向进给，来补偿在3D刀具半径补偿时的铣刀半径的偏差。在此，如果浸没深度ISD保持不变，那么平面不变，在这个平面上铣刀的端面不变。相对于一个标准刀具，带有例如：较小半径的铣刀到达不了也显示限制面积的槽底。对于一个自动的刀具横向进给，控制器必须已知限制面积。



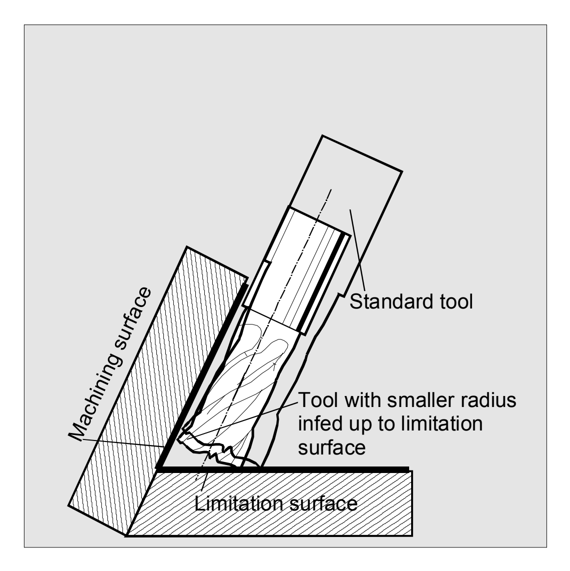
对于碰撞监控更多的信息参见

/PG/ 基础，“刀具补偿”

横向进给到限制面积CUT3DCCD的刀具中心点轨迹：（自软件版本SW6.4起）

如果使用一个和合适的标准刀具相比半径较小的刀具，那么一个纵向进给的铣刀会继续运行到再次碰到槽底。在此由加工面积和限制面积构成的角可以加工到刀具允许的大小。在此关系到圆周铣削和端铣的加工方式。

类似于缩小半径的刀具，在使用扩大半径的刀具时，沿相反方向相应的进给。





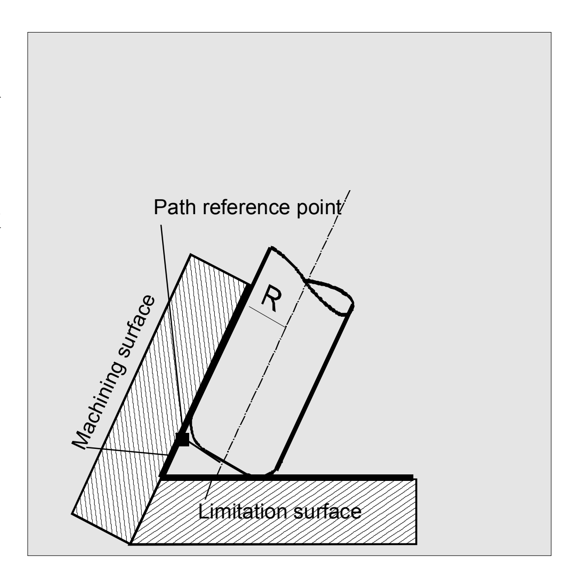
圆柱形刀具的使用

在使用圆柱形刀具时，只有当加工面积和限制面积构成一个锐角（小于90度）时，才需要横向进给。如果使用圆环铣刀（带刀尖圆弧的圆柱），那么铣刀不仅在锐角时而且在钝角时需要一个在刀具纵向的进给。

用CUT3DCC的3D-半径补偿：

（自软件版本SW 6.4起）在加工面积上的轮廓

如果用一个圆环铣刀的CUT3DCC当前有效，那么编程设计的轨迹和假定的相同直径的圆柱铣刀有关。这里生成的轨迹参考点在使用圆环铣刀时的情况如图所示。



在加工面积和限制面积之间的角度也可以在一个程序段内从锐角过渡到钝角，反之亦可。

相对于标准刀具，使用的实际刀具既可以大些，也可以小些。对此生成的角半径不可以是负的，并且生成的刀具半径的符号必须保持不变。

带刀尖圆弧的标准刀具

标准刀具的刀尖圆弧通过刀具参数$TC_DP7来描述。从刀具参数$TC_DP16中得出实际刀具相对于标准刀具的刀尖圆弧的偏差。

举例：

相对于标准刀具，减小半径的圆环铣刀的刀具尺寸

刀具类型	**R = 刀柄半径　r = 角半径**
带刀尖圆弧的标准刀具：	R = $TC_TP6　r = $TC_TP7
带刀尖圆弧的实际刀具： 刀具类型121和131圆环铣刀（带柄铣刀）	R’ = $TC_TP6 + $TC_TP15 + OFFN r’ = $TC_TP7 + $TC_TP6
在这个例子中 运用刀具类型($TC_DP1)。	不仅　$TC_TP15 + OFFN 而且　$TC_TP16　是负的。
只有带圆柱形刀柄的铣刀类型（圆柱铣刀或者带柄铣刀）以及圆环铣刀 （类型121和131） 和在极限情况下的圆柱形模具铣刀才可以。 （类型110）。 至软件版本SW6.4，带圆锥形刀柄 （类型155 – 157） 圆柱形刀柄和倒圆的刀尖的铣刀刀具是不允许的 （类型111）。	在使用这些允许的铣刀类型时，角半径r等于刀柄半径R。 其他所有允许的刀具类型都被说明为圆柱铣刀，并且不计算实际说明的刀尖圆弧的尺寸。

所有编号1–399的刀具类型都是允许的，
但是编号111和155至157例外。



工作流程

横向进给到限制面积CUT3DCCD的刀具中心点

轨迹：（自软件版本SW6.4起）

相对于G代码组22的所有其他刀具补偿，一个说明CUT3DCCD的刀具参数$TC_DP6对于刀具半径没有意义，并且不影响生成的补偿。补偿偏移从以下几个量中得出

- 磨损值
  (刀具参数 $TC_DP15)

和在限制面积上用于计算垂直偏移量的

- 可编程的偏移OFFN。

需要加工的面积在轨迹的左边或者右边，不可以从生成的零件程序中得知。这个可以从一个正向的半径和一个原始刀具负向的磨损值中得知。一个负向的磨损值总是描述一个缩小直径的刀具。

对于用G41，G42在当前有效的CUT3DCCD或者CUT3DCC时的刀具半径补偿必须有选项“定向转换”。

用CUT3DCC的3D-半径补偿：

（自软件版本SW 6.4起）在加工面积上的轮廓

在CUT3DCC时，NC零件程序和加工面积上的轮廓有关。在这种情况下，和通常刀具半径补偿一样得出总半径，这个总半径是由以下几个量组成的

- 刀具半径
  (刀具参数$TC_DP6)
- 磨损值
  (刀具参数 $TC_DP15)

和在限制面积上用于计算垂直偏移量的

- 可编程的偏移OFFN

限制面积的位置由两个值的分差决定

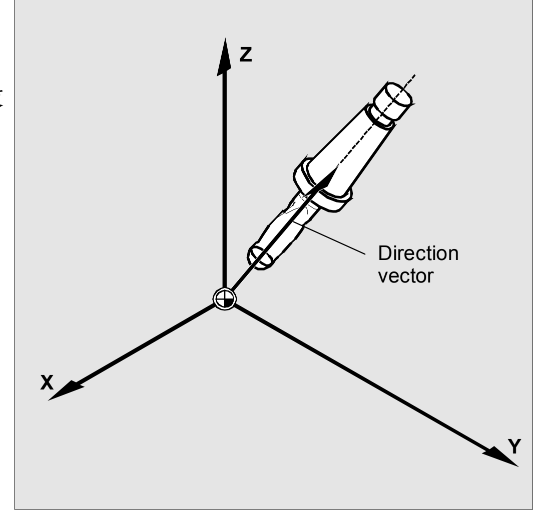
- 标准刀具的尺寸和
- 刀具半径(WZ-参数 $TC_DP6)

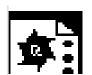
8.6 刀具定向, ORIC, ORID, OSOF, OSC, OSS, OSSE



刀具定向可以理解为在空间中刀具的几何取向。

使用一个5轴加工机床时刀具定向可以通过程序指令来设置。





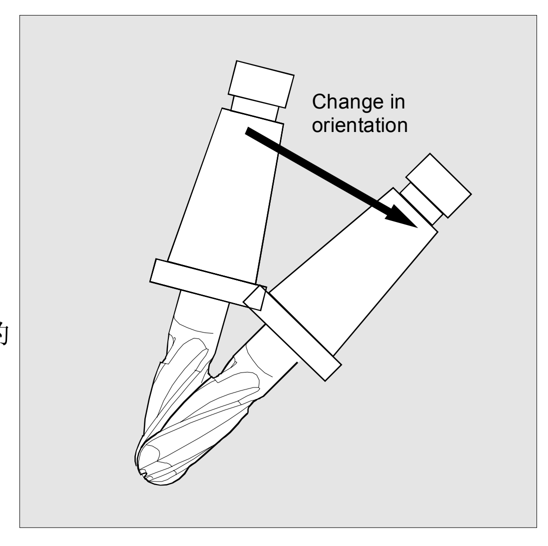
刀具定向编程

刀具定向的改变可以通过以下方式来编程：

- 回转轴的直接编程
- 欧拉或者RPY角
- 方向矢量
- LEAD/TILT(端铣)

参考坐标系要么是机床坐标系，(ORIMKS)要么是当前的工件坐标系(ORIWKS)。

定向改变可以受到以下影响：



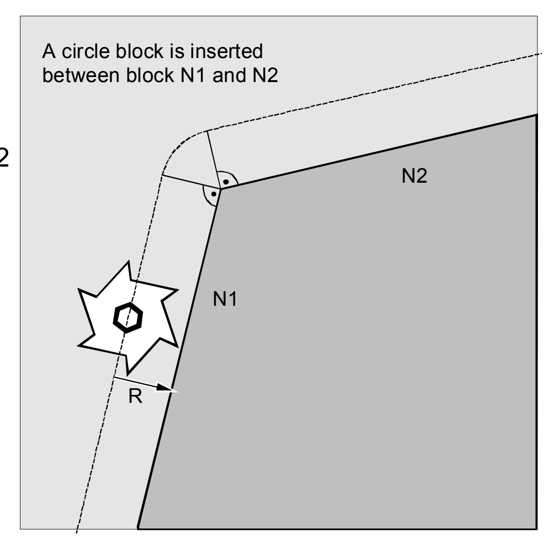
ORIC	定向和轨迹运动并行
ORID	定向和轨迹运动先后进行
OSOF	没有定向平整
OSC	定向恒定
OSS	只在程序段开始处的定向平整
OSSE	在程序段开始和结束处的定向平整
ORIS	在开通的定向平整时以度每毫米表示的定向改变的速度；适用于OSS和OSSE



外角处的特性

在外角处总是插入一个带铣刀半径的圆弧程序段。

用程序指令ORIC或者ORID可以确定，在程序段N1和N2之间编程的定向改变是在插入的圆弧程序段开始前执行还是和圆弧程序段同时执行。



如果在外角处需要一个定向改变，那么定向改变可以选择和插补并行或者和轨迹运动分离来进行。

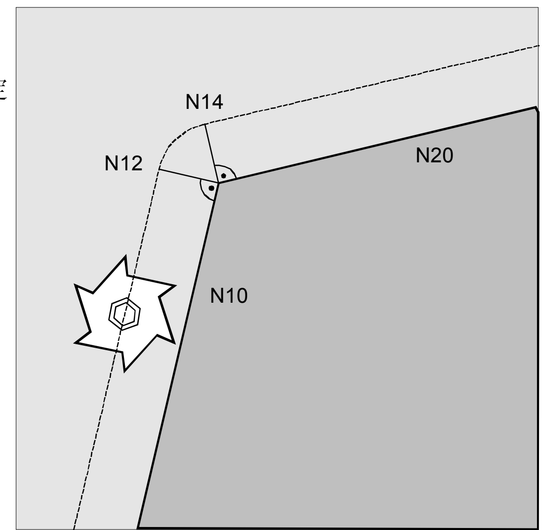
在ORID时，首先执行不带轨迹运动的插入的程序段。直接在两个运行程序段的第二个之前插入圆弧程序段，角是通过这两个程序段构成的。

如果在外角处插入几个定向程序段并且选中ORIC，那么圆弧运动和单独插入的程序段的定向改变的量相符，并且分布在这些程序段上。



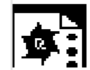
ORIC编程举例

如果在运行程序段N10和N20之间编程两个或者几个带定向改变（例如：A2=B2=C2=）的程序段，并且ORIC当前有效，那么插入的圆弧程序段和角度改变量相对应分布在这些中间程序段上。



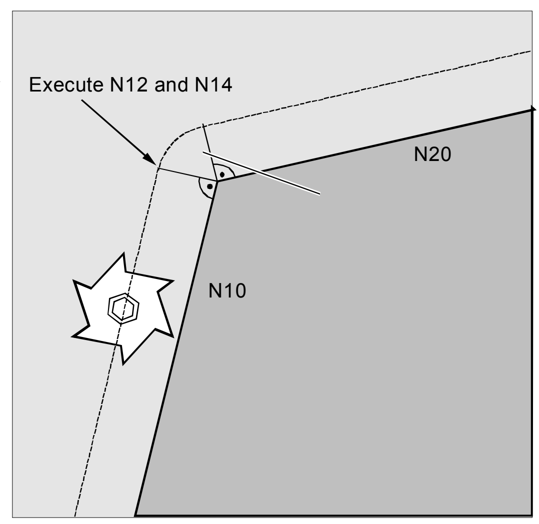
ORIC	
N8 A2=… B2=… C2=…	
N10 X… Y… Z…	
N12 C2=… B2=… N14 C2=… B2=…	在外角处插入的圆弧程序段分布在N12和N14，和定向改变相符。圆弧运动和定向改变在这种情况下并行处理。
N20 X =…Y=… Z=… G1 F200	



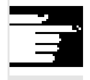


ORID编程举例

如果ORID当前有效，那么在两个运行程序段之间的所有程序段在第一个运行程序段结束时执行。带恒定方向的圆弧程序段在第二运行程序段之前直接执行。



ORID	
N8 A2=… B2=… C2=…	
N10 X… Y… Z…	
N12 A2=… B2=… C2=…	程序段N12和N14在结束处由N10来执行。之后运行带当前定向的圆弧程序段。
N14 M20	辅助功能等等
N20 X… Y… Z…	

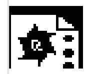


对于在外角处定向改变的方式，在外角的第一运行程序段中有效的程序指令是决定性的。



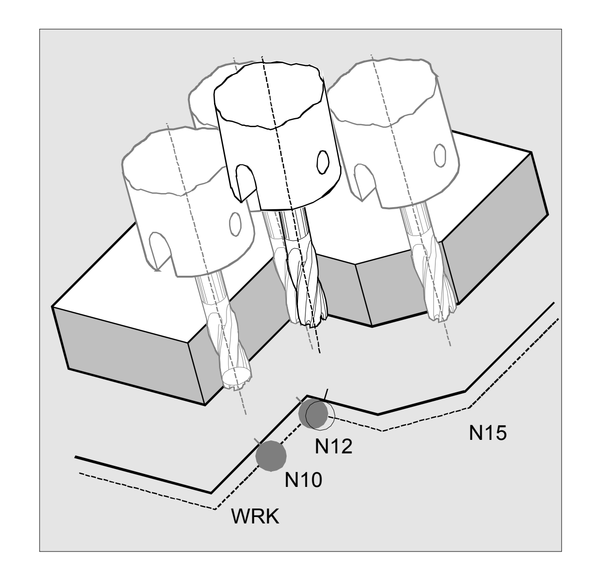
没有方向改变

如果不改变在程序段边界上的定向，那么刀具横截面是
一个圆弧，这个圆弧和两个轮廓相接触。



编程举例

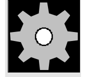
在一个内角处的定向改变




```
ORIC
N10 X …Y… Z… G1 F500
N12 X …Y… Z… A2=… B2=…, C2=…
N15 X Y Z A2 B2 C2
```

8.7 自由D编号分配，切削刃编号CE（自软件版本SW5起）

自软件版本SW5起可以使用D编号作为补偿编号。除此之外可以通过地址CE给切削刃的编号定址。
通过系统变量$TC_DPCE可以描述切削刃编号。
预设置：补偿编号== 切削刃编号
文献：FB，W1（刀具补偿）



机床制造商(MH 8.12)

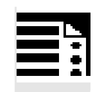
通过机床数据确定D编号的最大数量（切削刃编号）和每个刀具的最大切削刃数量。只有当确定最大切削刃编号（MD 18105）大于每个刀具（MD 18106）
的切削刃数量时，下面的指令才有意义。请注意机床制造商的说明。



其它说明

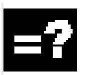
除了相关的D编号分配，也可以和T编号无关来分配D编号作为“平面的”或者“绝对的”D编号
（1-32000）（在功能“平面的D编号结构”内）。

8.7.1 检查D编号(CHKDNO)



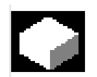
编程

```
state=CHKDNO(Tno1,Tno2,Dno)
```



参数说明

state	TRUE:	对于检查范围单一的分配D编号。
	FALSE:	产生一个D编号的重合或者给定参数无效。通过Tno1，Tno2和D编号来过渡导致重合的参数。这些数据可以在零件程序中计算。
CHKDNO (Tno1,Tno2)	检查命名刀具的所有D编号。	
CHKDNO(Tno1)	检查Tno1的所有D编号和其他所有的刀具比较。	
CHKDNO	检查所有刀具的所有D编号和其他所有刀具比较。	



功能

用CKKDNO可以检查现有的D编号是否是单一分配。

所有在一个TO单元内定义的刀具的D编号只能出现一次。

替代刀具在此不考虑。

8.7.2 重命名D-编号(GETDNO, SETDNO)

编程

```
d = GETDNO (t,ce)

state = SETDNO (t,ce,d)
```



参数说明

d	刀具刀刃的D编号
t	刀具的T编号
ce	刀具的刀刃编号（CE编号）
state	说明指令是否可以无误地执行（TRUE或者FALSE）。

功能

GETDNO

这个指令提供了一个带T编号t的刀具特定切削刃（ce)的D编号。

如果不存在输入参数的D编号，那么设置d=0。如果D编号无效，那么给回一个大于32000的值。

SETDNO

用这个指令可以给刀具t的切削刃ce的D编号值d赋值。通过state给回指令的结果（TRUE或者FALSE）

如果没有输入参数的数据程序段，那么给回FALSE。句法错误会导致报警。不可以明显将D编号设置为0。

举例：（重命名一个D编号）

```
$TC_DP2[1,2]  = 120
$TC_DP3[1,2]  = 5.5
$TC_DPCE[1,2] = 3;  切削编号CE
...
N10 def int DNrAlt, DNrNeu = 17
N20 DNrAlt = GETDNO(1,3)
N30 SETDNO(1,3,DNrNeu)
```

在此给切削刃CE=3赋值新的D值17。



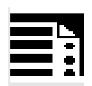
现在，切削刃的数据通过D编号17来触发；不仅可以通过系统变量，而且也可以在用NC地址的编程中。



其它说明

D编号必须单一分配。一个刀具的两个不同的切削刃不可以有同一个D编号。

8.7.3 求得预先给出D编号刀具的T编号（GETACTTD）



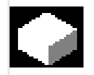
编程

```
status = GETACTTD (Tnr, Dnr)
```



参数说明

Dnr	D编号，针对D编号应寻找对应的T编号。
Tnr	找到的T编号
status	0: 找到T编号。Tnr包含T编号的值。 -1: 对于说明的D编号不存在T编号；Tnr=0。 -2: D编号不是绝对的。Tnr包含第一个找到的刀具的值，这个刀具包含带有值Dnr的D编号。 -5: 功能可以由于一个其他的原因不执行。



功能

用GETACTTD可以相对于一个绝对D编号，得出所属的T编号。不检查单一性。如果在一个TO单元内有几个相同的D编号，那么给回第一个找到的刀具的T编号。在使用“平面的”D编号时，使用指令是没什么意义的，因为这里总是给回值1（在数据保持中没有T编号）。

8.7.4 设置D编号无效



编程

DZERO



说明

DZERO	将TO单元的所有D编号标识为无效



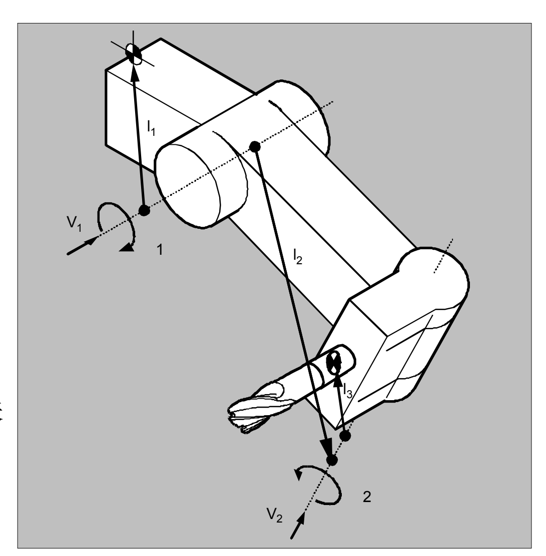
功能

这个指令在重新调整期间提供支持。
这样标记的补偿数据程序段不再由语言指令CHKDNO来检查。
为了使其可以重新使用，必须再次用SETDNO设置D编号。

8.8 刀架的运动关系

带最多两个旋转轴的刀架运动关系通过17系统变量$TC_CARR1[m]至$TC_CARR17[m]来描述。刀架的描述由以下几部分组成：

- 从第一旋转轴到刀架参考点的矢量距离l_1，从第一到第二旋转轴的矢量距离l_2，从第二旋转轴到刀具参考点的矢量距离l_3。
- 两个旋转轴的方向矢量 v_1, v_2。
- 围绕两个轴的旋转角度 α_1, α_2 。
  旋转角以视角方向沿旋转轴矢量的方向顺时针正向来计算。



删除的运动关系自软件版本5.3起

对于带有删除运动关系的机床
（刀具和工件都是可旋转的），系统变量扩展输入
$TC_CARR18[m] 至$TC_CARR23[m]，
并且描述如下：

可旋转的刀具台由以下几部分组成：

- 第二旋转轴v_2到第三旋转轴可旋转刀具台l_4的参考点的矢量距离。

回转轴由以下几项组成：

- 针对旋转轴v_1和v_2
  关系的两个通道名称，如果有可能可以在旋转轴的位置上在决定可定向刀架的定向时读取。

带有值T，P或者M其中之一的运动关系类型：

- 运动关系类型T：只有刀具是可旋转的。
- 运动关系类型P：只有工件是可旋转的。
- 运动关系类型M：刀具和工件是可旋转的。



可定向刀架系统变量的功能

名称	x-分量	y-分量	z-分量
l_1 偏移矢量	$TC_CARR1[m]	$TC_CARR2[m]	$TC_CARR3[m]
l_2 偏移矢量	$TC_CARR4[m]	$TC_CARR5[m]	$TC_CARR6[m]
v_1 旋转轴	$TC_CARR7[m]	$TC_CARR8[m]	$TC_CARR9[m]
v_2 旋转轴	$TC_CARR10[m]	$TC_CARR11[m]	$TC_CARR12[m]
α_1 旋转角 α_2 旋转角	$TC_CARR13[m] $TC_CARR14[m]		
l_3 偏移矢量	$TC_CARR15[m]	$TC_CARR16[m]	$TC_CARR17[m]
l_4 偏移矢量	$TC_CARR18[m]	$TC_CARR19[m]	$TC_CARR20[m]
旋转轴v_1和v_2的轴名称	旋转轴v_1和v_2的轴名称（无预先占用） $TC_CARR21[m] $TC_CARR22[m]		
运动关系类型 预保持T	$TC_CARR23[m]		
	运动关系类型-T ⇨	运动关系类型-P ⇨	运动关系类型-M
	只有刀具是工具可旋转的	只有工件是零件可旋转的	工件&刀具混合模式可旋转
旋转轴v_1的偏移 旋转轴v_2	旋转轴v_1和v_2以度表示的角度在接收基本状态时 $TC_CARR24[m] $TC_CARR25[m]		
旋转轴v_1和v_2的角度偏移	以旋转轴v_1和v_2的度数表示的切端面齿的偏移 $TC_CARR26[m] $TC_CARR27[m]		
旋转轴v_1和v_2的角度增量	以旋转轴v_1和v_2的度数表示的切端面齿的增量 $TC_CARR28[m] $TC_CARR29[m]		
旋转轴v_1和v_2的最小位置	旋转轴v_1和v_2 最小位置的软件极限 $TC_CARR32[m] $TC_CARR33[m]		
旋转轴v_1和v_2的最大位置	旋转轴v_1和v_2 最大位置的软件极限 $TC_CARR32[m] $TC_CARR33[m]		

旋转轴的参数 自软件版本SW 6.1起

系统变量扩展输入$TC_CARR24[m]至$TC_CARR33[m]，

并且描述如下：

旋转轴的**偏移**

- 在可定向刀架的基本状态下旋转轴v_1或者v_2的位置改变。

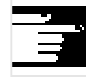
旋转轴的**角度偏移/角度增量**

- 旋转轴v_1和v_2的切端面齿的偏移或者角度增量。倒圆编程设计的或者计算出的角度到下一个值，这个值从来自公式phi = s + n * d的整数的n得出。

旋转轴的**最小位置/最大位置**

- 旋转轴v_1和v_2的极限角度（软件极限）。

可定向刀架的系统变量的扩展 自软件版本**SW 6.4**起			
名称	x-分量	y-分量	z-分量
刀架名称	代替一个数字，刀架可以获得一个名称。 $TC_CARR34[m]		
使用者： 轴名称1轴名称2 标记 定位	在使用者测量循环内预期的使用。 $TC_CARR35[m] $TC_CARR36[m] $TC_CARR37[m]		
	x-分量	y-分量	z-分量
	$TC_CARR38[m]	$TC_CARR39[m]	$TC_CARR40[m]
精偏移	可以添加到基础参数值里的参数		
l_1 偏移矢量	$TC_CARR41[m]	$TC_CARR42[m]	$TC_CARR43[m]
l_2 偏移矢量	$TC_CARR44[m]	$TC_CARR45[m]	$TC_CARR46[m]
l_3 偏移矢量	$TC_CARR55[m]	$TC_CARR56[m]	$TC_CARR57[m]
l_4 偏移矢量	$TC_CARR58[m]	$TC_CARR59[m]	$TC_CARR60[m]
v_1 旋转轴	$TC_CARR64[m]		
v_2 旋转轴	$TC_CARR65[m]		



其它说明

用"m" 分别说明所需描述刀架的编号。

轴上距离矢量的起始点和终点可以自由选择。围绕这两根轴的旋转角α_1, α_2

在刀架基本状态下用0°来定义。刀架的运动关系可以有任意多种可能性来描述。

只有一个旋转轴或者没有旋转轴的刀架可以通过一个或者两个旋转轴方向矢量的零设置来描述。在使用没有旋转轴的刀架时，距离矢量和附加的刀具补偿作用相同，刀具补偿的分量在进行加工平面（G17至G19）转换时不受影响。

删除刀架数据

用$TC_CARR1[0] = 0可以删除所有刀架数据程序段的数据。

自软件版本SW 5.3起

运动关系类型$TC_CARR23[T] = T
必须用三个所允许的大写或者小写字母（T，P，M）中的一个来覆盖，并且由于这个原因不可以被删除。

更改刀架数据

每个描述的值都可以通过零件程序中一个新的值的分配来改变。
每个其他不是T，P或者M的标志，在尝试激活可定向刀架时，会产生报警。

读取刀架数据

每个被描述的值可以通过对零件程序中变量的赋值来读取。

自软件版本SW6.4起

$TC_CARR34至$TC_CARR40
包含可供使用者自由支配的参数，并且这些参数至软件版本SW6.4按照标准在NCK内不再继续运用或者没有意义。

$TC_CARR41至$TC_CARR65 包含精偏移参数，
这些参数可以添加到基础参数的值里。如果将值40添加到参数编号中，那么得出从属于基础参数的精偏移值。

$TC_CARR47至$TC_CARR54以及
$TC_CARR61至$TC_CARR63
没有进行定义，并且在此后尝试读取或者读写时，会产生报警。

边界条件

刀架可以给一个刀具在所有可能的空间方向上定向，只有当

- 有两个旋转轴v_1和v_2。
- 旋转轴互相垂直。
- 刀具纵向轴垂直于第二旋转轴v_2。

自软件版本**SW 5.3**起

除此之外在使用机床时，在使用这些机床时所有可能的定向必须是可调节的，有下列要求：

- 刀具方向必须垂直于第一旋转轴v_1。

精偏移（自软件版本**SW6.4**起）

只有当一个可定向刀架被激活时，才可以识别一个允许的精偏移值，这个刀架包含一个这样的值并且同时设置数据SD42974是：TOCARR_FINE_CORRECTION = TRUE。

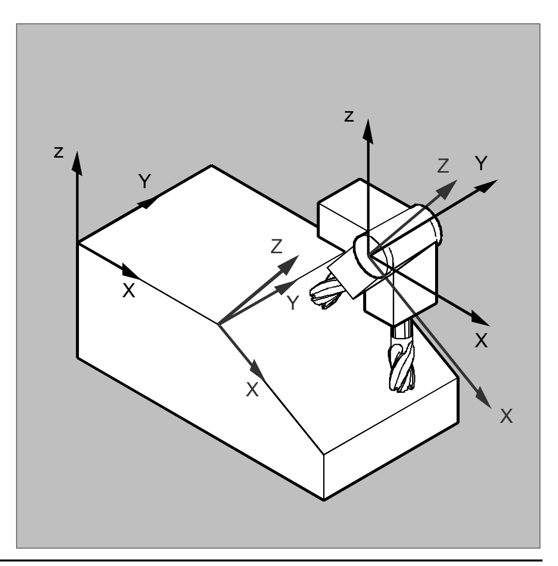
允许的精偏移的量可以通过机床数据限制在一个最大允许值上。





编程举例

在以下例子中使用刀架可以完全通过围绕Y轴的旋转来描述。



N10 $TC_CARR8[1]=1	刀架1的第一旋转轴Y向分量的定义
N20 $TC_DP1[1,1]=120	带柄铣刀的定义
N30 $TC_DP3[1,1]=20	长度20毫米
N40 $TC_DP6[1,1]=5	以及半径5毫米
N50 ROT Y37	以围绕Y轴旋转37°来定义框架
N60 X0 Y0 Z0 F10000	返回起始位置
N70 G42 CUT2DF TCOFR TCARR=1 T1 D1 X10	半径补偿，在旋转框架中设置刀具长度补偿，选择刀架1，刀具1
N80 X40	执行旋转37°的加工
N90 Y40	
N100 X0	
N110 Y0	
N120 M30	

■

轨迹位移性能



9.1 切向控制 TANG, TANGON, TANGOF, TLIFT, TANGDEL

编程

```
TANG (FAchse,Lachse1,LAchse2,Koppel,KS,Opt)
TANGON (FAchse,Winkel, Dist, Winkeltol)
TANGOF (FAchse)
TLIFT (FAchse)
TANGDEL (FAchse)
```

指令说明

TANG	预置的指令，用于定义一个切向的跟踪运行
	接通切向控制装置，说明跟随轴和偏移角度，可能情况下，精磨行程和角度偏差
TANGOF	关断切向控制系统，说明跟随轴
TLIFT	在轮廓拐角处添加中间程序段，TLIFT
	清除一个切向跟随运行的定义

参数说明

FAchse	跟随轴切向跟踪运行的附加回转轴
Lachse1, Lachse2	引导轴轨迹轴，确定跟踪的切向
Koppel	耦合系数切线角度的变化与跟踪轴之间的关系 参数选件；预调 1
KS	座标系统的标记字母 “B”= 基准坐标系； 可选择性参数； 预调 [“W” = 工件坐标系
Opt	优化： "S" 标准 （至软件版本SW6）缺省 "P" 自动适配到切向轴和轮廓的时间曲线
角度	跟随轴的偏移角
Dist	跟随轴的磨削行程，在Opt“P“时要求
Winkeltol	跟随轴的角度公差，（选件）， 仅在Opt=“P“时求值





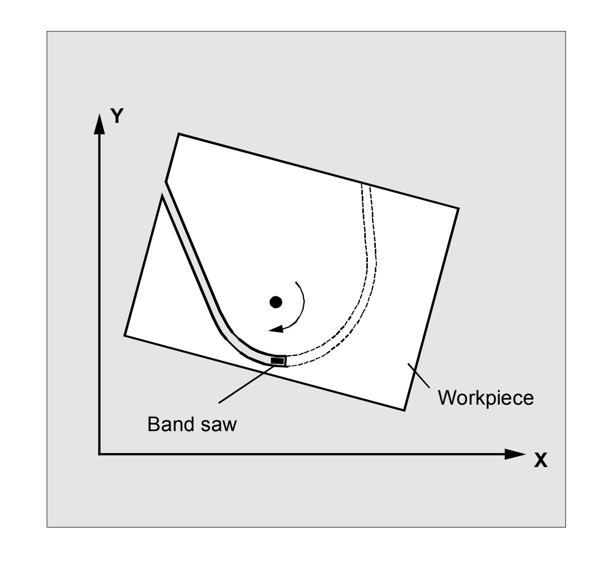
功能

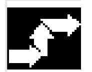
按照引导轴确定的轨迹的切线，跟随轴跟踪运行。由此一个刀具可以调整到与轮廓平行。通过TANGON指令中编程的角度，刀具可以与切线相关调用。

应用范围

切向控制此外可用于：

- 步冲时可转动的刀具切向调整
- 在使用锯条时工件矫正的跟踪（见图）
- 砂轮上修整器的调整（见下图）玻璃或纸张加工
- 用的小切削轮的调整
- 5轴焊接时的切向输入线材





工作流程

定义跟随轴和引导轴

用TANG定义跟随轴和引导轴。耦合系数表明切线角度变化和被跟踪轴之间的关系。通常其数值为1（预调）。跟踪可以在基准坐标系统“B“（预调）中或工件坐标系“W”中进行。

举例：

```
TANG(C,X,Y,1,"B")
```

表示：

回转轴C跟随着几何轴X和Y。

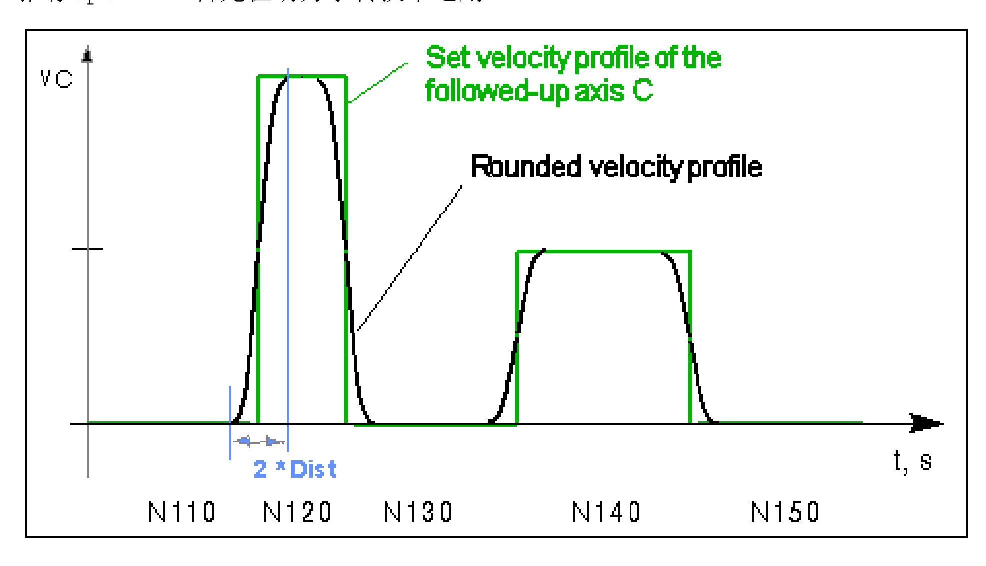
自软件版本SW7起，优化方式：

在Opt="P"时，引导轴的速度限制要一起考虑跟随轴的动态性能。

参数（距离和角度公差）限制跟踪轴和引导轴的切线间的误差。由于引导轴轮廓的变化而引起的跟随轴的速度变化，通过（距离和角度公差）平滑。

在此将跟随轴带预见性地导入（见图表），以使误差尽可能减到最小。

推荐Opt＝"P"首先在动力学转换中运用。



简化编程：

耦合系数1无需精确编程。

TANG(C, X, Y, 1, "B", "P") 可以作为

TANG(C, X, Y, , , "P")简化标记。到目前为止，

TANG(C, X, Y, 1, "B", "S") 可以作为

TANG(C, X, Y)写入。

TLIFT(…)指令在轴分配之后，用TANG(…)说明。举例：

```
TANG(C,X,Y…)
TLIFT(C)
```

TLIFT关断

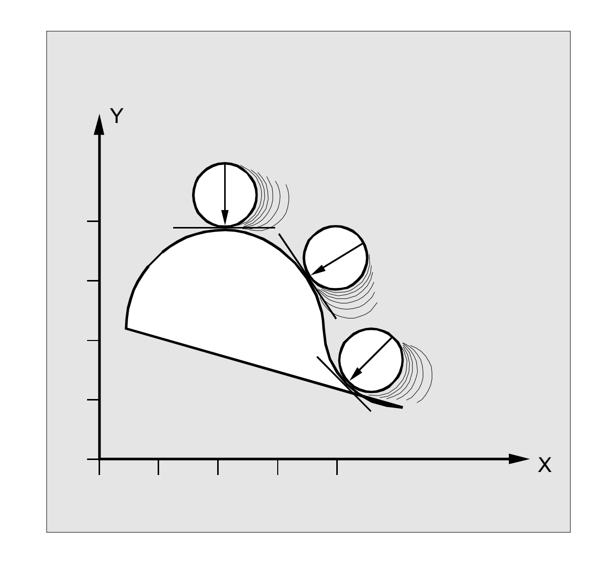
在此重复轴分配 TANG(…)，不带跟随的TLIFT(…)。

接通/关闭切向控制装置:

TANGON, TANGOF

使用TANGON，说明跟随轴和所要求地跟随轴
TANGON(C,90）的偏移角度，调用切向控制装置:

```
TANGON(C,90)
```



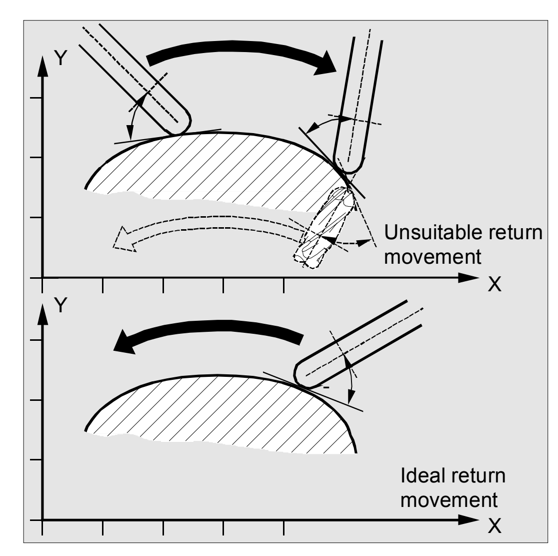
表示：
C轴是跟随轴。它在每次轨迹轴的运动中向轨迹切线转动90度位置。

切向控制系统的关断，说明跟随轴：

```
TANGOF(C)
```

工作范围限制中的临界角

在来回导入的轨迹运动中，切线在轨迹换向点上转向180度， 相应地跟随轴的矫正发生改变。
通常这样的行为无效回返运动在同样的负偏移角中，和正向运动一样开始。



对此将限制跟随轴的工作范围。工作范围限制在轨迹换向的时间点上，须是活性的。 (WALIMON)

当偏移角在工作范围限制之外时，尝试将负偏移角重新带入容许的工作范围内。

软件7：

在Opt＝“P”时，此功能无效。达到工作范围限制后，在这里形成警报输出的停止。



其它说明

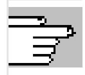
在轮廓拐角处添加中间程序段，TLIFT

在轮廓拐角处切线改变，从而被跟踪的轴的额定位置变得不稳定。轴在通常情况下，尝试以其可能的最大速度平衡这种变化在此情况下在轮廓上拐角下，相对于所期望的切向定位，出现超过一定距离的误差。如这些因为工艺原因不被容许的话，
TLIFT指令将促使控制系统，在拐角处停止并在一自动生成的中间程序段中将被跟踪的轴旋转到新的切线方向。

如跟随轴某一次被作为轨迹轴运行，则编程的轨迹轴旋转。通过 FGREF[ax] = 0.001功能，
被跟踪轴的速度在这里将达到最大值。

到现在为止被跟踪轴没有被作为轨迹轴，而是作为定位轴处理。这样，速度依赖于由机床数据确定的定位速度。

被跟踪轴进行最大速度转动



自从插入一个中间程序段开始，角度变化由机床数据定义
$MA_EPS_TLIFT_TANG_STEP。

清除一个切向跟随运行程序的定义

如果用一个同样的跟随轴，在预备调用TANG中定义的一个新的切向运行，则必须清除一个已存在的用户定义的切向运行。

`TANGDEL(Fachse)`	清除切向跟随
清除只能在TANGOF(Fachse)的耦合被关闭的情况下进行	

编程举例

平面转换实例

`N10 TANG(A, X, Y,1)`	1. Tang的定义激活耦合的跟踪运行
`N20 TANGON(A)`	
`N30 X10 Y20`	
`...`	
`N80 TANGOF(A)`	关闭第一个耦合
`N90 TANGDEL(A)`	删除第一个定义
`...`	2. Tang.跟随运行定义，激活新的耦合
`TANG(A, X, Z)`	
`TANGON(A)`	
`...`	
`N200 M30`	



编程举例

带几何轴转换和TANGDEL

	不产生报警。
`N10 GEOAX(2,Y1)`	Y1是几何轴2
`N20 TANG(A, X, Y)`	
`N30 TANGON(A, 90)`	
`N40 G2 F8000 X0 Y0 I0 J50`	
`N50 TANGOF(A)`	
`N60 TANGDEL(A)`	用Y1去除跟踪运行的激活
`N70 GEOAX(2, Y2)`	一号定义的清除
`N80 TANG(A, X, Y)`	Y2是新的几何轴2
`N90 TANGON(A, 90)`	2.Tang的定义。
...	跟随运行。激活第2个定义的跟踪运行



编程举例

带自动跟踪运行，带距离和角度公差的自动优化

```
N80 G0 C0
N100 F=50000
N110 G1 X1000 Y500
N120 TRAORI                          ；带轴向公差的精磨削
N130 G642
N171 TRANS X-1200 Y-550              ；轨迹速度的自动优化
N180 TANG(C,X,Y, 1,,"P")             ；精磨削行程5mm
N190 TANGON(C, 0, 5.0, 2.0)          ；角度公差2度
N210 G1 X1310 Y500
N215 G1 X1420 Y500
N220 G3 X1500 Y580 I=AC(1420)_
                     J=AC(580)
N230 G1 X1500 Y760
N240 G3 X1360 Y900 I=AC(1360)_
                     J=AC(760)
N250 G1 X1000 Y900
N280 TANGOF(C)
N290 TRAFOOF
N300 M02
```

其它说明

对转换的影响

被跟踪的回转轴位置可以成为用于转换的输入值。

跟随轴的明确定位

如一跟随引导轴运行的跟随轴明确定位，则定位参数附加影响编程的偏移角。

允许所有的位移规定：轨迹轴和定位轴运行。

耦合的状态

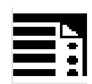
在NC零件程序中可以用以下系统变量询问耦合的状态：

`$AA_COUP_ACT[轴]`

0　　无激活的耦合

1,2,3　切向跟踪运行有效

9.2 联动 TRAILON, TRAILOF



编程

```
TRAILON(Fachse,LAchse,Koppel)
TRAILOF(Fachse,Lachse, Achse2)
```

指令和参数说明

TRAILON	激活和定义联动组合；模态有效
TRAILOF	关断联动组合
	联动轴的轴名称
LAchse	引导轴的名称
Koppel	耦合系数＝ 联动轴行程/ 引导轴行程 预设置＝1



功能

在一个已定义的引导轴的运动中，它所属的联动轴（＝跟随轴）运行引导轴引导的移动位移，同时考虑到一个耦合系数。

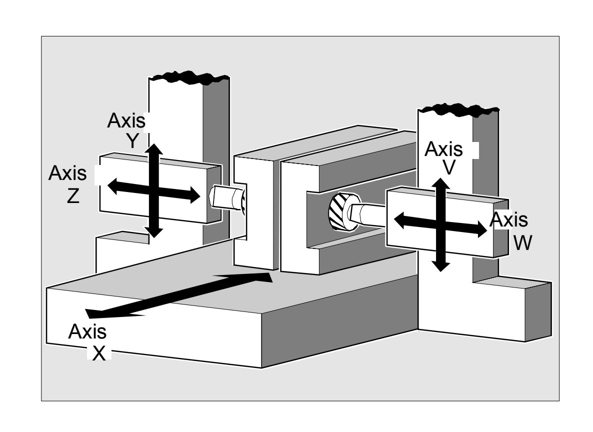
引导轴和跟随轴共同组成联动轴组合。

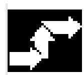
应用范围

- 通过一个模拟轴进行轴运行。引导轴是一个模拟轴，联动轴是一个真正的轴。真实轴运行，考虑到耦合系数。
- 两个联动轴组合的双面加工：

  1. 引导轴 Y;联动轴 V

  2. 引导轴 Z，联动轴 W





工作流程

联动组合 定义, TRAILON

用模态有效的语言指令TRAILON同时定义和开通一个联动组合。

```
TRAILON(V,Y)
```

V = 联动轴, Y = 引导轴

可同时激活的联动组合的数量只由机床上所存在的轴的组合可能性限制。



联动始终在基准坐标系（BKS)中进行。

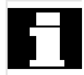
关联的轴的类型

联动组合由线性轴和回转轴任意组合构成。一个模拟轴也可在此被定义为引导轴。

联动轴

一个联动轴最多可同时被两个引导轴分配。这种分配通过不同的联动组合实现。

一个联动轴可以用所有可支配的运动指令编程（G0, G1,G2,G3,…)。此外与已定义位移无关的是，联动轴运行其耦合系数从引导轴导出的位移。



一个联动轴也可以是其余联动轴的引导轴。以这种方式可以建立不同的联动组合。

耦合系数耦合系数说明了联动轴和引导轴位移之间的要求关系。

$$\text{耦合系数} = \frac{\text{联动轴的行程}}{\text{引导轴的行程}}$$

如在编程中耦合系数未被说明，则耦合系数1自动有效。

带小数点的数据将作为分数输入（TYPE REAL）。
输入负数值，表明引导轴和联动轴在相反方向运行。

关断联动组合
通过语言指令关闭引导轴的耦合。

```
TRAILOF(V,Y)
```

V = 联动轴, Y = 引导轴
有两个参数的TRAILOF只关闭第一个引导轴的耦合。

一个联动轴有两个引导轴。例如， **V=联动轴,X,Y=** 引导轴，可用作关闭有**3**个参数的耦合**TRAILOF:**

```
TRAILOF(V,X,Y)
```



其它说明

加速度和速度

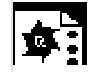
属于耦合的轴的加速度极限和速度极限由联动组合中“最弱的轴”决定。

耦合的状态

在NC零件程序中可以用以下系统变量询问耦合的状态：

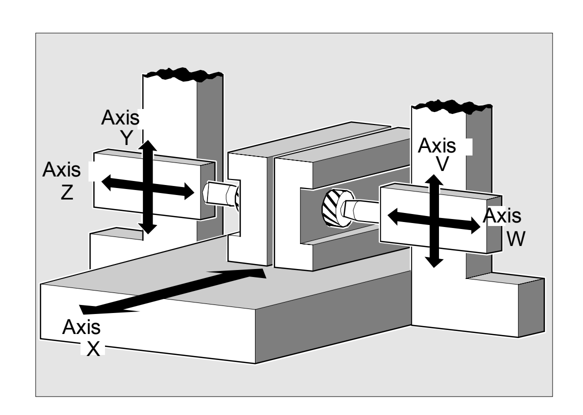
$AA_COUP_ACT[Achse]

0	无有效的耦合
8	联动轴有效



编程举例

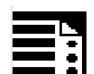
工件两面须根据展示的轴的情况加工。据此构成两个联动组合。



...	
`N100 TRAILON(V,Y)`	接通第一个联动组合
`N110 TRAILON(W,Z,-1)`	接通第二个联动组合，耦合系数是负：联动轴运动方向与其各自对应的引导轴相反
`N120 G0 Z10`	Z-,W-轴的在相反方向的横向进给
`N130 G0 Y20`	Y-,V-轴的在相同轴向的横向进给
...	
`N200 G1 Y22 V25 F200`	联动轴V的有关或无关的运动的叠加
...	
`TRAILOF(V,Y)`	关断第一个联动组合
`TRAILOF(W,Z)`	关断第二个联动组合

9.3 曲线表 CTABDEF, CTABEND, CTABDEL, CTAB, CTABINV/CTABFNO

9.3.1 带曲线表的语言指令 CTABID, CTABLOCK, CTABUNLOCK



编程

带曲线表的模态有效的语言指令：

A) 重点功能：
曲线图表的定义在零件程序中实现。在语言指令单的最后您可以找到参数说明。

`CTABDEF (FAchse,LAchse,n,applim, memType)`	定义曲线图表开始。
`CTABEND ()`	定义曲线图表末尾。
`CTABDEL ()`	删除所有曲线图表，不考虑存储器类型（自软件程序6.3起）。
B) 形式(自软件版本 **SW6.3**起):	在存储器SRAM或DRAM中：
`CTABDEL (n, m, memType)`	删除存储在mem类型中的曲线图表范围中的曲线图表。
`CTABNOMEM (memType)     (SW 6.4 以上)`	已定义的曲线图表的数目
`CTABFNO (memType)       (SW 6.4 以上)`	还有可能的图表的数目
`CTABID (n, memType)     (SW 6.4 以上)`	提供在存储器类型中作为第n个曲线图表登记的图表序号。
`CTABLOCK (n, m, memType)`	设置对删除和改写的禁止
`CTABUNLOCK (n, m, memType)`	取消对删除和改写的禁止 CTABUNLOCK使能用CTABLOCK禁止的图表。在激活的耦合中有效的图表继续保持禁止状态并不能被删除。只要通过使激活的耦合失效取消禁止，CTABLOCK的禁止就失效。 从而可以删除此图表。再次CTABUNLOCK调用是不必要的。

C)其它形式的应用 (自软件版本 SW 6.3起)	**可选择的参数，用于选择**
`CTABDEL(n)`	删除一个曲线图表。
`CTABDEL(n, m)`	删除一个曲线图表范围。
`CTABDEL(, , memType)`	删除规定的存储器中的所有曲线图表。
`CTABLOCK(n)`	禁止删除和改写 n号的曲线图表
`CTABLOCK(n, m)`	禁止从n到m范围的曲线图表。
`CTABLOCK()`	所有已存在的曲线图表。
`CTABLOCK(, , memType)`	所有在存储器类型中的曲线图表。
`CTABUNLOCK(n)`	取消删除和改写n号的曲线图表。
`CTABUNLOCK(n, m)`	再次使能从n导m号码范围的曲线图表。
`CTABUNLOCK()`	所有已存在的曲线图表。
`CTABUNLOCK(, , memType)`	所有在存储器类型中的曲线图表。

D)其他形式的运用（自软件程序6.4起）	**轴耦合的诊断**
`CTABID (n, memType)` `CTABID(p, memType)`	提供第n和p号的带mem类型存储器的曲线图表。
`CTABID(n)`	提供第n号的曲线图表，通过 MD 20905:确定的存储器类型。
`CTABISLOCK(n)`	送回n号的曲线图表的禁止状态。
`CTABEXIST (n)`	检测n号的曲线图表。
`CTABMEMTYP (n)`	送回设有n号曲线图表的存储器。
`CTABPERIOD (n)`	送回曲线图表的周期性。
`CTABSEG (memType)`	已使用的曲线段在相关存储器类型中的数目。
`CTABSEGID (n)`	使用n号的曲线图表的数量。曲线段。
`CTABFSEG(memType)`	可能的曲线段的数目。
`CTABMSEG(memType)`	最大可能的曲线段数量。
`CTABPOLID (n)`	使用带n号的曲线图表的数目。
`CTABFPOL(memType)`	在相关的存储器类型中还可能的曲线多项式数量。
`CTABMPOL  (memType)`	在相关存储器类型里最大可能的曲线多项式的数目。

从曲线图表**CTAB,CTABINV**中所求得的跟随轴和引导轴位置。 （自软件程序－标准**5.1**起）

`R10=CTAB (LW,n,grad,FAchse,LAchse)`	跟随值对一个引导值
`R10=CTABINV (FW,aproxLW,n,grad,FAchse,LAchse)`	引导值对一个跟随值

通过引导值的说明，用**CTABSSV,CTABSEV**确定曲线图表的段。 （自软件版本**6.3**起）

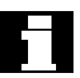
`R10=CTABSSV(LW,n,grad,FAchse,LAchse)`	属于LW段的跟随轴的起始值
`R10=CTABSEV(LW,n,grad,FAchse,LAchse)`	属于LW段的跟随轴的终值

在一个曲线图表的起始和终端点，跟随轴和引导轴的数值**CTABTSV,CTABTEV,CTABTSP,CTABTEP.** （自软件版本**6.4**起）

`R10=CTABTSV (n,grad,FAchse)`	曲线图表起始的跟随值
`R10=CTABTEV (n,grad,FAchse)`	曲线图表起始的跟随值
`R10=CTABTSP (n,grad,LAchse)`	曲线图表起始的引导值
`R10=CTABTEP (n,grad,LAchse)`	曲线图表末端的引导值

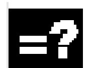
跟随值的曲线图表的值范围
CTABTMIN, CTABTMAX

`R10=CTABTMIN (n,FAchse)`	整个周期中曲线图表最小的跟随值
`R10=CTABTMAX (n,FAchse)`	整个周期中曲线图表最大的跟随值
`R10=CTABTMIN(n, a, b, FAchse, LAchse)`	曲线图表上、在a…b引导值范围内的最小的跟随值
`R10=CTABTMAX(n, a, b, FAchse, LAchse)`	在曲线图上、在a…b引导值范围内的最大的跟随值



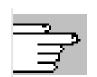
取消在图表定义范围内的R参数的赋值。
举例：

```
...
R10=5 R11=20
...
CTABDEF
G1 X=10 Y=20 F1000
R10=R11+5      ;R10=25
X=R10
CTABEND
...            ;R10=5
```



参数说明

FAchse	跟随轴 跟随轴是通过曲线图表编程的轴。
LAchse	引导轴 通过引导轴编程引导值。
R10	参数名 参数名，在此参数名下应存储起始值和终值。
LW	引导值 引导轴的位置值，由此计算出一个跟随值。
FW	引导值跟随轴的位置值，由此计算出一个引导值。
n, m	曲线图表的序号；n<m 例如 在CTABDEL(n，m） 曲线图表的序号是明确的，并且与存储器类型无关。在SRAM和DRAM中不能存在相同序号的图表。
a, b	引导值的范围（在CTABMIN和CTABMAX中可选择说明） 在这个范围中a和b两个界限都被要求。在引导值的曲线图表的定义范围外的数值中，取引导值的两个极限值的中的一个。
grad	上升参数的参数名。
p	登记处（在存储区mem类型）。
applim	图表周期性的标志： 0 非周期性图表 图表1周期性与引导轴相关 图表2周期性与引导轴和跟随轴相关
aproxLW	如不能明确确定一个跟随值的引导值，为跟随值的近似值。
FAchse,LAchse	随动轴或引导轴的可选择性参数。
memType	NC存储器类型的可选择性参数："DRAM" / "SRAM" 如对这些参数没有编程数值，则使用通过CTAB_DEFAULT_MEMORY_TYPE设置的标准存储器类型。



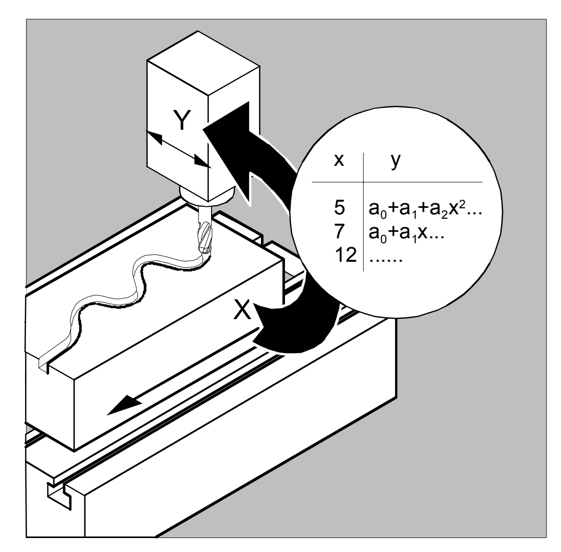
关于引导值和跟随值参见本章“轴向引导值耦合”部分。

存在对轴耦合和优化资源使用的诊断的其他功能。其余信息见此
/FB/,M3，轴耦合和ESR



功能

可以借助曲线图表编程两轴之间的位置和速度关系。

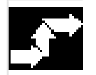


例如，置换机械的曲线字盘：通过实现引导值和跟随值之间的
功能性关联，曲线图表由此构成轴向引导值耦合的基础。
在相应的编程时，从相互所属的引导轴和跟随轴的位置中，控制系统计算出一个与曲线字盘相应的多项式。

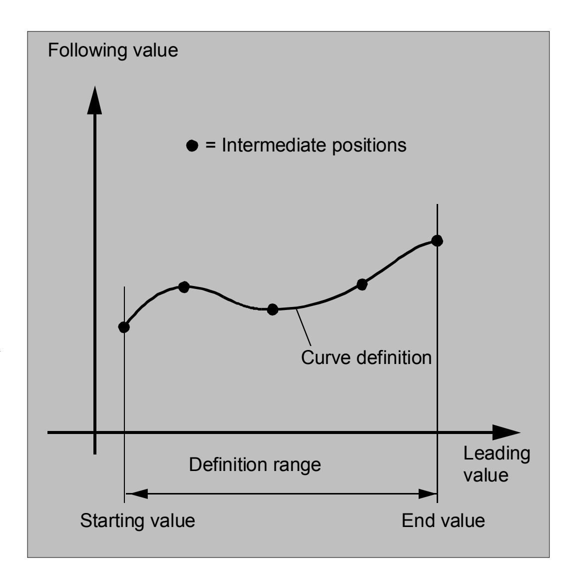


其它说明

对于曲线图表的建立，必须保留关于机床数据安装的存储器位置。



曲线图表的定义：
CTABDEF, CTABEND



一个曲线图表显示一个零件程序或一个零件程序段，
此程序或程序段通过前面加CTABDEF和结束指令
CTABEND进行标识。

在此程序段范围内，通过运动指令，使引导轴的各个位置明确地分配到跟随轴，这些跟随轴位置作为计算曲线中的各个点以多项式的形式（最大为3级多项式）
使用 。

自软件版本SW6起，曲线的支点以多项式形式（最大为5级）计算。

曲线图表的始值和终值：

在曲线图表定义内，相关轴位置的第一个参数作为对曲线图表定义范围的开始的始值（第一个运动指令）曲线图表的定义范围的终值相应地由最后的运行指令决定。

在曲线图表的定义内，可使用整个NC语言范围。

所有在曲线图表范围内碰到的模态有效的指令，在图表定义结束时失效。进行图表定义的零件程序，在图表定义的前后处于相同的状态。



其它说明

不允许：

- 进刀停止
- 引导轴运动中的跳跃（例如，转换的转换）
- 单独的跟随轴的运动指令
- 引导轴的运动换向，即引导轴的位置必须始终明确。
- 不同程序级的CTABDEF和 CTABEND指令

ASPLINE, BSPLINE, CSPLINE的激活

如在曲线线图表 CTABDEF（）. . .，CTABEND中，一个ASPLINE,BSPLINE或者CSPINE被激活，则在激活样条前至少编程一个起始点。应该避免在CTABDEF后立即激活，因为否则样条由曲线图表定义之前的实际轴位置决定。

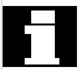
举例：

```
...
CATBDEF(Y, X, 1, 0)
X0 Y0
ASPLINE
X=5 Y=10
X10 Y40
...
CTABEND
```



自软件版本**SW6.3**起

与机床数据MD 20900：
CTAB_ENABLE_NO_LEADMOTION决定，
在引导轴没有运动时可以有跟随轴的跳跃。其他在说明中提到的限制继续有效。
在图表的设置和删除中可以使用NC存储器类型的参数。



曲线图表的重复使用
如果表格存储在静止存储器（SRAM)中，则通过曲线图表计算出的、引导轴和跟随轴的功能性关系保持在所选择的图表号之下，与零件程序结束和关机无关。
在动态存储器（DRAM）中设置的图表在关机后删除，如有可能须再次生成。一次设置的曲线图表可以在任意引导轴和跟随轴的轴组合中使用，这些轴用于生成曲线图表。

曲线图表的删除, **CTABDEL**
曲线图表可用CTABDEL删除。在轴耦合中有效的曲线图表不能被删除。如果自多次删除命令CTABDEL或者CTABDEL（n.m）的曲线图表至少有一个有效,则不删除已设定地址的曲线图表。
自软件版本SW6.3起，可以通过可选的存储器参数删除一个确定存储器类型的曲线图表。

通过“执行外部程序”装载曲线图表
在执行外部曲线图表时，必须选择后装载缓存器的大小，通过MD18360:MM_EXT_PROG_BUFFER_SIZE进行选择，从而使整个曲线图表定义可以同时存储在后装载缓存器中。
否则警报15050将停止零件程序的加工。

曲线图表的改写

只要一个新曲线图表定义时其序号被使用，这个曲线图表将被改写。

例外：一个曲线图表在一个轴耦合中有效或者由CTABLOCK()禁止。



其它说明

- 在曲线图表的改写中不给出相应的警告。
- 使用系统变量$P_CTABDEF，可以在任何时候由零件程序进行询问：曲线图表定义是否有效。
- 在排除曲线图表定义的指令后，此零件程序段可以作为实际零件程序重新运用。

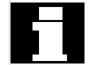


编程举例

CTABSSV和CTABSEV的应用

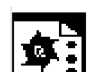
不改变一个程序段，用于定义一个曲线图表。只要程序段没有用于图表定义，则进刀停止STOPRE用的指令保持不变，并且立即生效。CTABDEF 和CTABEND 被取消：

```
CTABDEF(Y,X,1,1)
...
...
IF NOT ($P_CTABDEF)
STOPRE
ENDIF
...
...
CTABEND
```



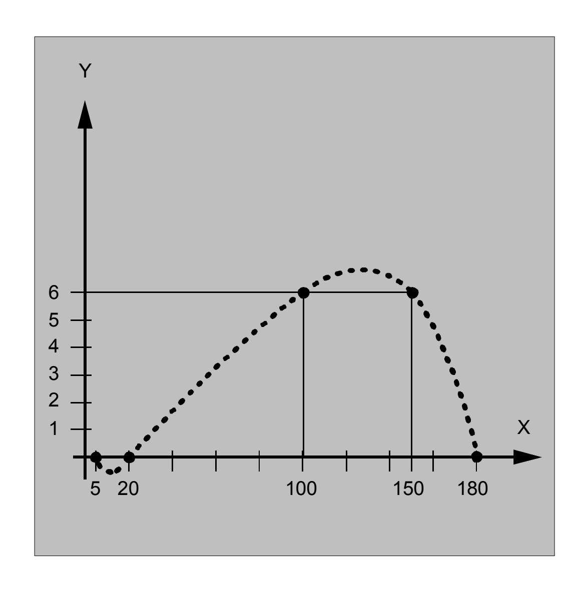
曲线图表和不同的运行状态

在有效的程序段搜索时，曲线图表的计算是不可能的。如目标程序段在曲线图表的定义内，在达到CTABEND时将给出警报。



编程举例1

曲线图表的定义



N100 CTABDEF(Y,X,3,0)	非周期性的曲线图表3定义的开始
N110 X0 Y0	1. 运动指令，确定起始值和第一支点： 引导值：0; 跟随值： 0
N120 X20 Y0	2. 支点：引导值：0…20; 跟随值： 起始值：0
N130 X100 Y6	3. 支点：引导值：20…100; 跟随值: 0…6
N140 X150 Y6	4. 支点：引导值：100…150; 跟随值： 6…6
N150 X180 Y0	5. 支点：引导值：150…180; 跟随值： 6…0
N200 CTABEND	定义末尾；曲线图表在其内部生成为最大3级的多项式；带给定支点的曲线，其计算与模态选择的插补方式有关（圆弧一，线形一，样条插补）；定义开始前的零件程序状态被恢复。



编程举例2

一个周期性的曲线图表2定义，引导值范围从0到360，
跟随轴运动从0到45，然后回到0：

N10 DEF REAL DEPPOS;	
N20 DEF REAL GRADIENT;	
N30 CTABDEF(Y,X,2,1)	定义的开始
N40 G1 X=0 Y=0	
N50 POLY	
N60 PO[X]=(45.0)	
N70 PO[X]=(90.0) PO[Y]=(45.0,135.0,-90)	
N80 PO[X]=(270.0)	
N90 PO[X]=(315.0) PO[Y]=(0.0,-135.0,90)	
N100 PO[X]=(360.0)	
N110 CTABEND	定义的结束

通过耦合Y到X对曲线进行测试：

```
N120 G1 F1000 X0
N130 LEADON(Y,X,2)
N140 X360
N150 X0
N160 LEADOF(Y,X)
```

在引导值75.0时读图表功能：

```
N170 DEPPOS=CTAB(75.0,2,GRADIENT)
```

引导轴和跟随轴的定位：

```
N180 G0 X75 Y=DEPPOS
```

在耦合开通后，跟随轴的同步是不必要的：

```
N190 LEADON(Y,X,2)
N200 G1 X110 F1000
N210 LEADOF(Y,X)
N220 M30
```

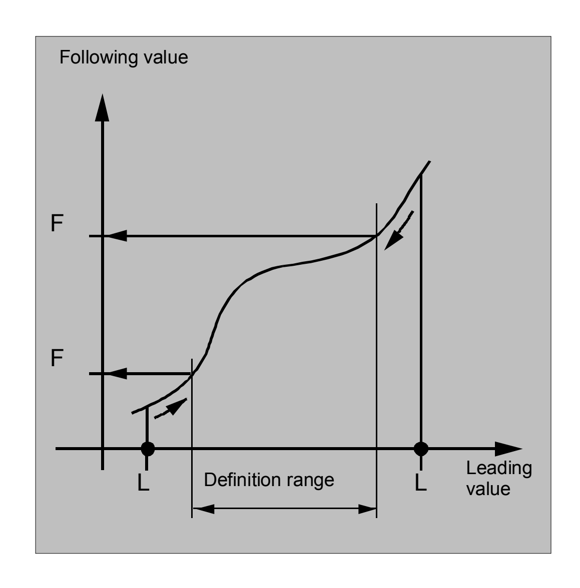
9.3.2 曲线图表的边缘性能 CTABTSV/CTABTSP, CTABTMIN

起始边界值和终值： **CTABTSV, CTABTEV, CTABTSP, CTABTEP (自软件版本 SW 6.4起)**

值范围，最小和最大： **CTABTMIN, CTABTMAX (自软件版本 SW 6.4起)**

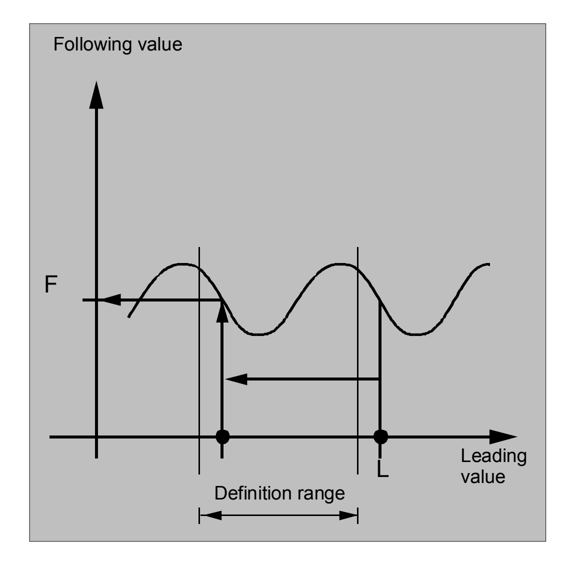
非周期的曲线图表

如引导值在定义范围外，上限或下限将作为跟随值给出。



周期性的曲线图表

如引导值在定义范围之外，则取模计算定义范围内的引导值，并输出相应的跟随值。



读取曲线图表的边缘值 CTABTSV, CTABTEV, CTABTSP, CTABTEP

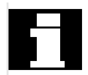
使用CTABTSV可以由一个跟随值读曲线图表开始处的值。
使用CTABTSV可以在曲线图表的末尾读取跟随轴的值。

使用CTABTSV可以由引导轴读曲线图表开始处的值。
使用CTABTEP可在曲线图表的末尾读取引导轴的值。

一个曲线图表的始值和终值与其用上升或是下降的引导值定义无关。始值总由范围下限，终值总由范围上限确定。

CTABTMIN, CTABTMAX

使用 CTABTMIN 和CTABTMAX曲线图表的最小或最大范围可以在整个区域或是一个被定义的范围内确定。对引导值的相应的范围将给出两个界限。



CTABTSV, CTABTEV, CTABTSP, CTABTEP, CTABTMIN, CTABMAX的语言指令可以

- 由零件程序或
- 由同步动作直接

使用。

- 内部的功能实现的时间CTABINV( ) CTABINV( )有关。
- CTABTSV, CTABTEV, CTABTSP, CTABTEP （CTABTMIN, CTABTMAX只在引导值没有给出范围的情况下

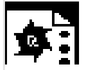
图表分段的数量无关。

在同步动作中读取

用户在同步动作中使用指令 CTABINV( )或
CTABTMIN( )和 CTABTMAX( )时应注意：
在执行时刻

- 有足够的NC功率可以使用，或者
- 在调用前询问曲线图表的段的数量，从而可以划分相关的图表。

其余的有关编程同步动作的部分将在第十章中描述。



编程举例

CTABTSV, CTABTEV, CTABTSP, CTABTEP, CTABTMIN, CTABMAX的应用

曲线图表的最小和最大值的确定。

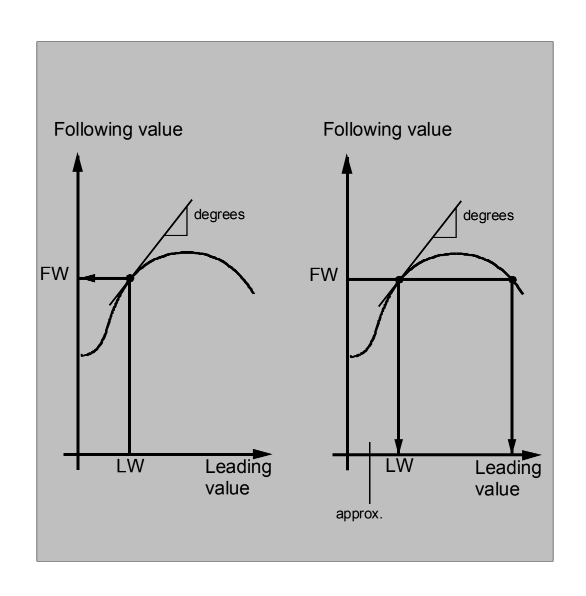
```
N10 DEF REAL STARTVAL
N20 DEF REAL ENDVAL
N30 DEF REAL STARTPARA
N40 DEF REAL ENDPARA
N50 DEF REAL MINVAL
N60 DEF REAL MAXVAL
N70 DEF REAL GRADIENT
...
```

`N100 CTABDEF(Y,X,1,0)`	图表定义的开始
`N110 X0 Y10`	第一起始值图表段第一
`N120 X30 Y40`	图表段终值＝第二起图表段起始值
`N130 X60 Y5`	
`N140 X70 Y30`	
`N150 X80 Y20`	
`N160 CTABEND`	图表定义的末尾
`...`	起始位置 STARTPOS = 10，
`N200 STARTPOS = CTABTSV(1, GRADIENT)`	终止位置ENDPOS= 图表中的20，
`N210 ENDPOS = CTABTEV(1, GRADIENT)`	以及 STARTPARA = 10, ENDPARA =
`N220 SRARTPARA = CTABTSP(1, GRADIENT)`	80 从跟随轴的数值范围内读取。
`N230 ENDPARA = CTABTEP(1, GRADIENT)`	
`...`	
`N240 MINVAL = CTABTMIN(1)`	Y = 5 时的最小值，和
`N250 MAXVAL = CTABTMAX(1)`	Y = 40 时的最大值。

9.3.3 对曲线图表位置和图表分段的存取 CTAB, CTABINV

读图表位置, CTAB, CTABINV

使用CTAB可以由零件程序或者同步动作（章节10）直接读取引导值的跟随值。



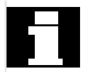
使用CTABINV可以读取跟随值的引导值。这种分配不要求始终明确。CTABINV 由此需要一个近似值 (aproxLW)，用于所期待的引导值。CTABINV 送回与近似值最接近的引导值。.近似值可以是前一个插补节拍的引导值。

两个功能在相应的位置另外输出图表功能的上升值给上升参数（度）。这样，引导轴或者跟随轴的速度可以在相应的位置计算。

读曲线段位置, CTABSSV, CTABSEV
(自软件版本SW 6.3起)

使用CTABSSV可以读给定引导值曲线段的起始值。

使用CTABSSV可以读给定引导值曲线段的终点值。



语言指令CTAB,CTABINV和 CTABSSV, CTABSEV 可以

- 由零件程序或者
- 由同步动作直接

使用。

所有与同步动作相关的编程均在第10章中介绍。

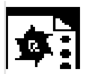


其它说明

如果引导轴和跟随轴使用不同的长度单位设计，则引导轴或者跟随轴的可选参数说明在CTAB/CTABINV/CTABSSV/CTABSEV时就很重要。

在以下情况下，语言指令CTABSSV和CTABSEV必须询问不适合的编程分段：

1. 圆弧或者渐开线编程。
2. 棱边或者圆角带 CHF、 RND 有效。
3. 带G643的精磨削有效。
4. 压缩器比如用 COMPON, COMPCURV, COMPCAD 有效。



编程举例

CTABSSV和CTABSEV的应用

确定属于引导值X=30的曲线段。

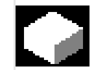
```
N10 DEF REAL STARTPOS
N20 DEF REAL ENDPOS
N30 DEF REAL GRADIENT
...
```

程序	注释
N100 CTABDEF(Y,X,1,0)	图表定义的开始
N110 X0 Y0	第一个图表段的起始位置
N120 X20 Y10	第一个图表段的终点位置＝第二个图表段的起始位置…
N130 X40 Y40	
N140 X60 Y10	
N150 X80 Y0	
N160 CTABEND	图表定义的末尾
...	
N200 STARTPOS = CTABSSV(30.0, 1, GRADIENT)	第二段的起始位置Y＝10
...	
N210 ENDPOS = CTABSEV(30.0, 1, GRADIENT)	第二段的终止位置Y＝40 第二段属于LW=30.0

9.4 轴向引导值耦合, LEADON, LEADOF

编程

```
LEADON(FAchse,LAchse,n)
LEADOF(FAchses,LAchse,n)
```

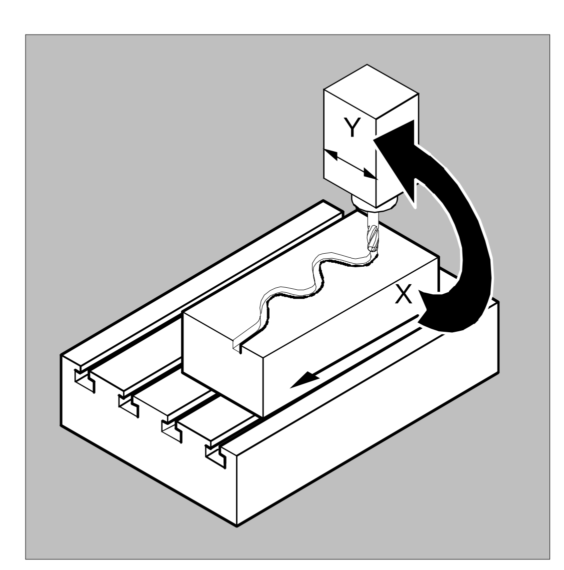


说明

LEADON	接通引导值耦合
LEADOF	关断引导值耦合
FAchse	跟随轴
LAchse	引导轴
n	曲线图表编号

功能

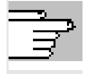
在轴的引导值耦合时同步运行一个引导轴和一个跟随轴。因此跟随轴的位置通过曲线图表或由此计算出的多项式明确地分配到一个引导轴的位置，也可以模拟。



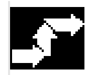
引导轴即提供曲线图表的输入值的轴。**跟随轴**即接受曲线图表上所有已计算出的位置的轴。

引导值耦合既通过零件程序又可以在同步动作时(第十章)的运动接通和关断。

引导值耦合一直在基准坐标系中有效。



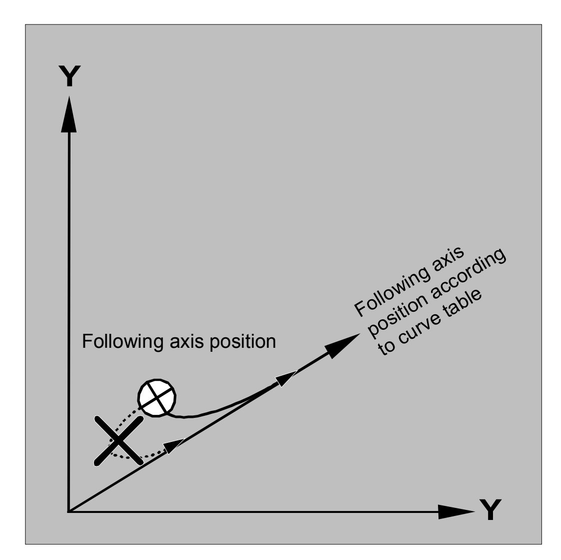
曲线图表的完成见本章中的“曲线图表”段。
引导值耦合
/FB/, M3, 联动　和引导值耦合。



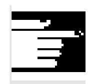
工作流程

引导值耦合要求引导轴和跟随轴的同步动作。只有跟随轴在根据曲线图表计算出的曲线的公差范围内，引导值耦合打开的情况下，才能达到同步动作。

跟随轴位置的公差范围是根据机床数据 37200 COUPLE_POS_TOL_COARSE定义的。

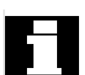


如跟随轴在引导值耦合接通时还不位于相应的位置上，同步运动自动产生，只要计算所得的跟随轴位置的额定值与其实际的跟随轴位置接近。这时在同步动作期间，跟随轴在跟随轴给定速度定义地方向运行（由引导轴速度和CTAB计算出）。



其它说明

如计算所得的、在引导值耦合打开时跟随轴的额定位置与其实际跟随轴位置相距很远，则不产生同步运动。



实际值和额定值耦合

作为引导值，即用于计算位置的输出值可以使用：

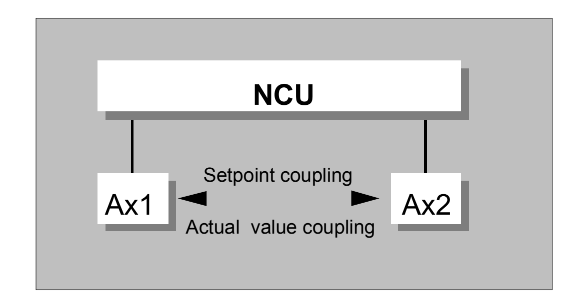
- 引导轴位置的实际值：实际值耦合
- 引导轴位置的额定值：额定值耦合

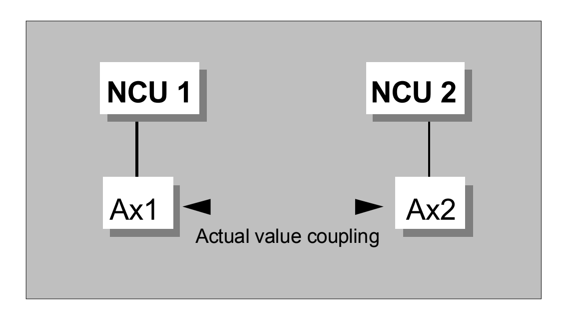


其它说明

与实际值耦合相比，额定值耦合提供更好的引导轴和跟随轴之间的同步运动，因此符合预置符合标准。

只有当引导轴和跟随轴插补由相同的NCU进行时，才可能进行额定值耦合。一个外部的引导轴上仅可以通过实际值，使跟随轴与引导轴耦合。





实际值和额定值之间的**转换**

只可以通过设定数据 $SA_LEAD_TYPE
进行一次转换。

实际值和额定值间的转换应一直在跟随轴静止状态下完成。因为只有在静止状态时，转换后才能重新同步。

应用示例：

额定值的读取在大的机床振荡时不能完成。在启动引导值耦合时，如在压力传输时有大的振荡，则有必要从实际值耦合切换到额定值耦合。

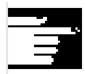


额定值耦合时的引导值模拟

对于引导轴，插补器通过机床数据与伺服分开。从而额定值耦合时额定值可以在引导轴不运动的情况下得到。

通过额定值耦合得到的额定值，可以在以下变量同步动作的使用中读出。

- $AA_LEAD_P 引导值位置
- $AA_LEAD_V 引导值速度



其它说明

引导值可有选择性地在其他自动编程的程序中产生。这样产生的引导值将写入变量

- $AA_LEAD_SP 引导值位置
- $AA_LEAD_SV 引导值速度

并从中读取。为使用这些变量，必须采用设定数据
$SA_LEAD_TYPE = 2

耦合的状态

在NC零件程序中可以用以下系统变量询问耦合的状态：$AA_COUP_ACT[轴]

0	无耦合有效
16	引导值耦合有效

关闭引导值耦合， LEADOF

如关闭引导轴耦合，跟随轴可重新成为正常的指令轴！

轴向引导值耦合和不同的运行状态

与机床参数的设定有关，引导值耦合由RESET关断。



编程举例

在冲压设备中，在一个引导轴（冲杆轴）和由传输轴与辅助轴构成的传输系统的轴之间，这种传统地机械耦合应由一个电子的耦合系统代替。

这表明了，在一个冲压设备中一个机械的传输系统如何被一个电子的传输系统所代替。耦合和去耦的过程作为静止的同步动作被实现。

传输轴和辅助轴可通过曲线图表作为跟随轴被引导轴LW(冲杆轴)控制。

跟随轴

X	进给轴或者纵向轴
YL	结束轴或者横向轴
ZL	冲程轴
U	轧辊进给，辅助轴
V	定向头，辅助轴
W	润滑，辅助轴

状态管理

开关和耦合过程通过实时变量：

$AC_MARKER[i]=n

管理，使用:

i	标记号
n	状态值

作用

作为出现在同步动作中的动作，例如：

- 耦合，LEADON（跟随轴，引导轴，曲线图表号码）
- 输出耦合，LEADOF（跟随轴，引导轴）
- 设定实际值，PRESETON（轴，值）
- 设置标记器, $AC_MARKER[i]= 值
- 耦合方式：实际/虚拟的引导值
- 轴位置的返回，POS[轴]＝值

条件

作为条件，与逻辑运算符“与” 连接在一起的数字快速输入、实时变量和位置比较将被处理。

提示

在以下例子中使用换行、首行缩格和加粗字体，只为提高编程的可读性。为了控制，所有在标志行数下的都占一行。

注释

定义所有的静止的同步动作

; **** 复位标记器

```
N2      $AC_MARKER[0]=0 $AC_MARKER[1]=0
        $AC_MARKER[2]=0 $AC_MARKER[3]=0
        $AC_MARKER[4]=0 $AC_MARKER[5]=0
        $AC_MARKER[6]=0 $AC_MARKER[7]=0
```

; **** E2 0=>1 打开耦合传输

```
N10     IDS=1        EVERY ($A_IN[1]==1) AND
($A_IN[16]==1) AND ($AC_MARKER[0]==0)
DO      LEADON(X,LW,1) LEADON(YL,LW,2)
        LEADON(ZL,LW,3) $AC_MARKER[0]=1
```

; **** E2 0=>1 打开耦合轧辊进给

```
N20     IDS=11       EVERY ($A_IN[1]==1) AND
($A_IN[5]==0) AND ($AC_MARKER[5]==0)
DO      LEADON(U,LW,4) PRESETON(U,0)
        $AC_MARKER[5]=1
```

; **** E2 0=>1 打开定位头耦合

```
N21     IDS=12       EVERY ($A_IN[1]==1) AND
($A_IN[5]==0) AND ($AC_MARKER[6]==0)
DO      LEADON(V,LW,4) PRESETON(V,0)
        $AC_MARKER[6]=1
```

; **** E2 0=>1 打开润滑耦合

```
N22     IDS=13       EVERY ($A_IN[1]==1) AND
($A_IN[5]==0) AND ($AC_MARKER[7]==0)
DO      LEADON(W,LW,4) PRESETON(W,0)
        $AC_MARKER[7]=1
```

; **** E2 0=>1 耦合 “关”

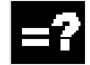
```
N30     IDS=3  EVERY ($A_IN[2]==1)
DO      LEADOF(X,LW) LEADOF(YL,LW)
        LEADOF(ZL,LW) LEADOF(U,LW)
LEADOF(V,LW) LEADOF(W,LW) $AC_MARKER[0]=0
$AC_MARKER[1]=0 $AC_MARKER[3]=0
$AC_MARKER[4]=0 $AC_MARKER[5]=0
$AC_MARKER[6]=0 $AC_MARKER[7]=0
....
N110    G04 F01
N120    M30
```

9.5 进给运行, FNORM, FLIN, FCUB, FPO



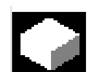
编程

```
F… FNORM
F… FLIN
F… FCUB
F=FPO(…,…,…)
```



说明

FNORM	基本设置。进给值通过程序段的轨迹行程来规定，并且作为模态值有效。
FLIN	线性的轨迹速度剖面图： 进给值由实际值通过轨迹行程线性的、由程序开始到程序末尾运行，其后作为模态值有效。这种性能可与G93和G94联系在一起。
FCUB	立体的轨迹速度剖面图： 以程序段方式编程的F值－与程序段结束处有关－通过一个样条连接。样条以切线形式与前一个或者后一个进给说明开始和结束，与G93和G94一起作用。 如在一个程序段中缺少F地址，在这里将使用最后编程的F值。
F=FPO…	有关多项式的轨迹速度剖面图： F地址通过从一个实际值到程序段末尾的的多项式说明进给过程。此后终值作为模态值有效。



功能

对可变的进给进程的给定值，进给编程将根据DIN66025在线性的和空间的进程上扩大。空间的进程可以直接或者作为插补的样条被编程。
由此，与须加工的工件的曲线有关，平滑的速度进程被连续地编程。
速度的过程使可没有冲击的速度变化成为可能，并通过这完成均匀的工件表面加工。

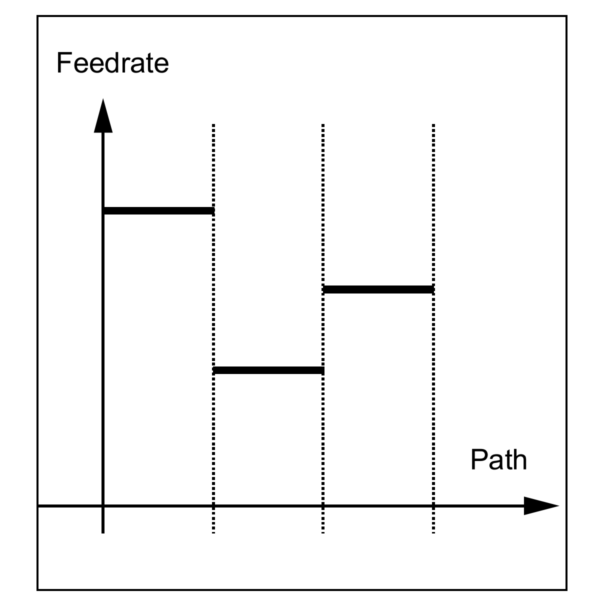


工作流程

FNORM

进给地址F把轨迹进给表示为符合DIN66025的恒定值。

更多的信息可以在编程说明“基础部分”中找到。

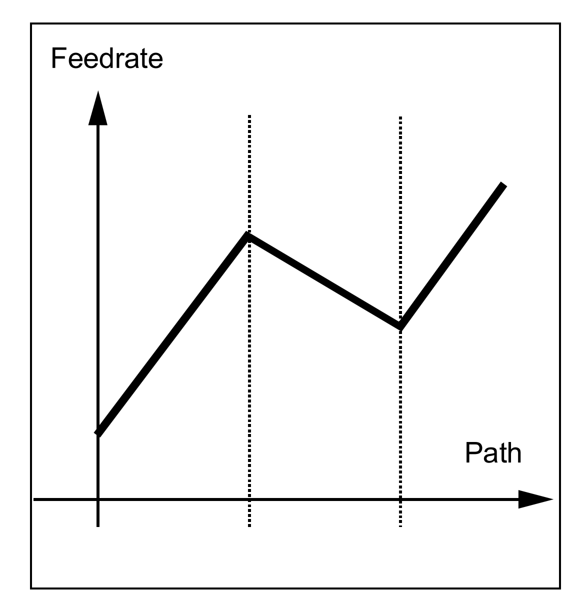


FLIN

进给进程通过实际的进给值到被编程的F值，线性运行到程序段末尾。

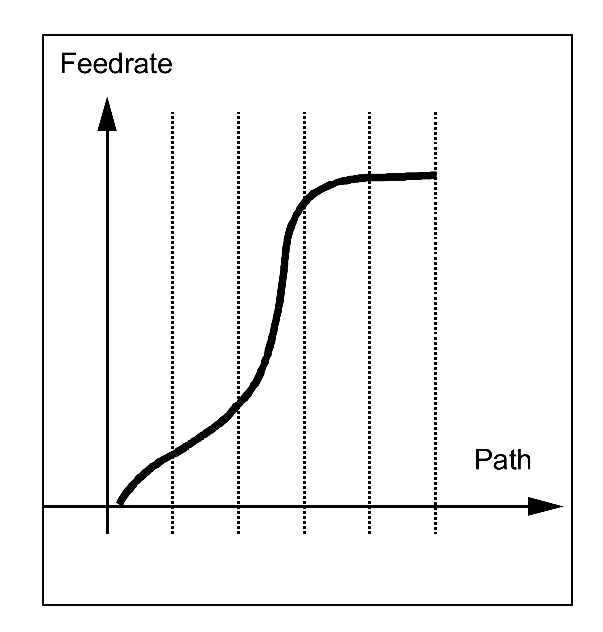
举例：

```
N30 F1400 FLIN X50
```



FCUB

从实际的进给值到编程的F值到程序段结尾，进给以空间的行程运行。控制系统通过样条连接所有有效FCUB程序方式编程的进给值。进给值作为计算样条插补的支点。



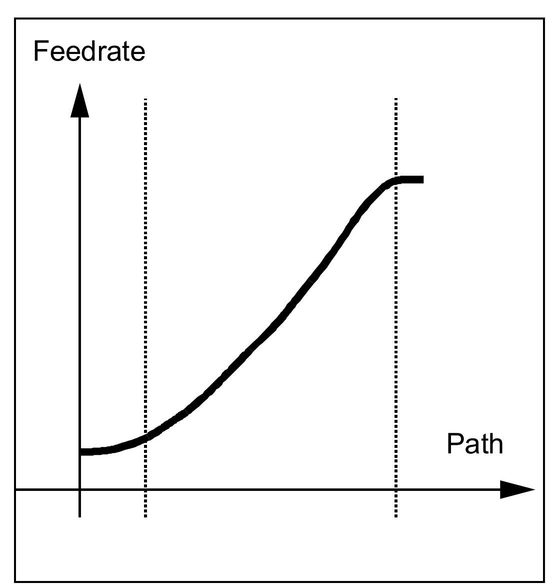
举例：

```
N50 F1400 FCUB X50
N60 F2000 X47
N70 F3800 X52
…
```



F=FPO(…,…,…)

进给进程通过一个多项式直接编程。多项式系数的参数类似于多项式插补。



举例：

```
F=FPO(endfeed, quadf, cubf)
```

endfeed, quadf 和 cubf 是前面所定义的变量。

endfeed:	程序段结束时的进给
quadf:	二次方的多项式系数
cubf:	立方的多项式系数

在有效的FUCB中，样条在程序段开始和末端，切向与FPO确定的进程相连。

边界条件

与编程的进给进程无关，轨迹运行方式的编程的功能有效。

可编程的进给进程原则上绝对有效－与G90或G91无关。



进给进程FLIN和FCUB与G93和G94共同作用。

FLIN和FCUB不与G95, G96/G961 和 G97/G971共同作用。



其它说明

压缩器

在有效的压缩器COMPON中，
把几个程序段汇编成一个样条段：

FNORM:
对于样条段，最后的程序段F字有效。

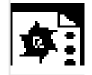
FLIN:
对于样条段，最后的程序段F字有效。
编程的F值在段的末尾有效，并以线性返回。

FCUB:
所产生的进给样条与机床数据
$MC_COMPESS_VELO_TOL
中所定义的值偏差最大，与所编程的终点偏离。

F=FPO(…,…,…)
程序段不压缩。

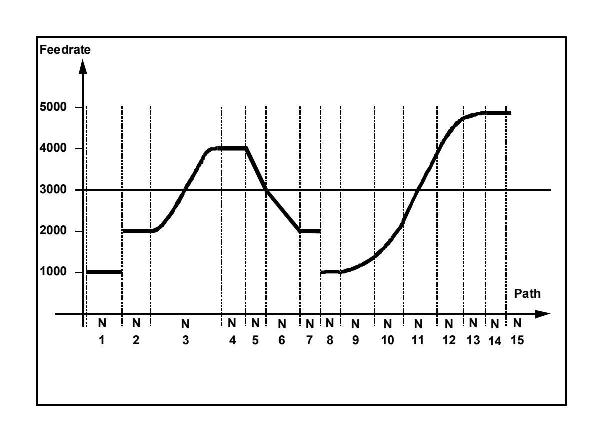
在弯曲的轨迹中的进给优化

进给多项式F=FPO和进给样条FCUB应始终以恒定切削速度CFC进行。由此得出一个加速度恒定的额定进给剖面图。



编程举例

在这个例子中可找到不同进给剖面图的编程和图示。



N1 F1000 FNORM G1 X8 G91 G64	恒定的进给剖面图，增量尺寸。
N2 F2000 X7	跳跃的额定速度改变
N3 F=FPO(4000, 6000, -4000)	通过多项式定义的进给轮廓，速度4000，在程序段末尾
N4 X6	多项式进给4000被看作模态值
N5 F3000 FLIN X5	线性的进给轮廓
N6 F2000 X8	线性的进给轮廓
N7 X5	线性的进给被看作模态值
N8 F1000 FNORM X5	有跳跃的速度变化的恒定的进给轮廓
N9 F1400 FCUB X8	所有以下以程序段方式编程的F值与样条联系在一起。
N10 F2200 X6	
N11 F3900 X7	
N12 F4600 X7	
N13 F4900 X5	关闭样条轮廓
N14 FNORM X5	
N15 X20	

9.6 带进刀存储器的程序过程, STARTFIFO, STOPFIFO, STOPRE

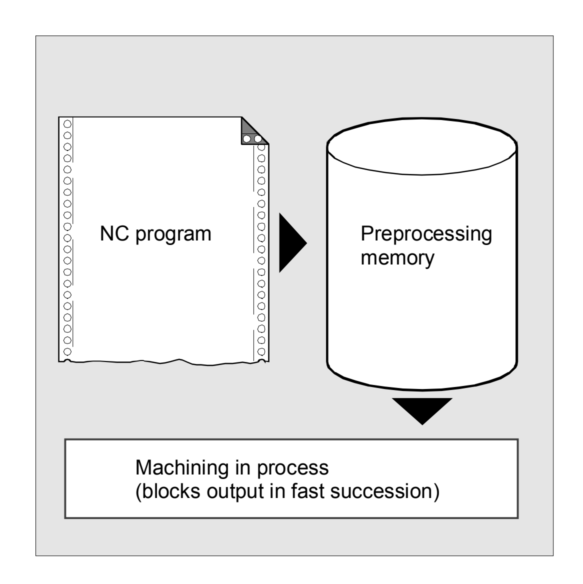


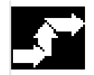
指令说明

STOPFIFO	停止快速加工区段，装载进刀存储器 直至 STARTFIFO, "进刀存储器满" 或者“程序段结束”为止。
STARTFIFO	快速加工段的开始，与进刀缓冲器的装载并行。
STOPRE	进刀停

功能

根据扩展级，控制系统具有一定数量的进刀存储器，它们在加工完成前已经存储预处理的程序段，并在加工过程中给出快速的程序段序列。
由此，可以以高速度运行较短的位移。
只要控制系统的剩余时间允许，进刀存储器将被装载。
直到进给存储器被装载或STARTFIFO或STOPRE被识别，STOPFIFO停止加工。





工作流程

标识加工段

应在进刀存储器中被存储的加工段，在STOPFIFO或STARTFIFO的开始或末尾会被标识。

举例：

```
N10 STOPFIFO
N20…
N100
N110 STARTFIFO
```

在进刀存储器被装载或在STARTFIFO命令之后，程序段的加工处理才开始。

限制

进刀存储器的装载不会被执行或中断，如果加工段包含着命令，这些命令迫使一个未被缓冲的运行。（参考点运行，测量功能，…)

停止进刀

在STOPRE的编程中，只有把所有先前准备和存储的程序段完全加工后，跟踪的程序段才会被执行。前面的程序段停止在准确停（如同G9）

举例：

```
N10 …
N30 MEAW=1 G1 F1000 X100 Y100 Z50
N40 STOPRE
```

在读取机床状态数据时($A…)，控制器生成内部进刀停。

举例：

R10 = $AA_IM[X]	；读取X轴的实际值



提示

在刀具补偿生效时并且在样条插补时，不应编程STOPRE，因为否则属于一个整体的程序段顺序就会中断。

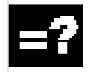
9.7 条件中断的程序段, DELAYFSTON, DELAYFSTOF

条件中断的零件程序段被称作停止延迟区。



编程

`N... DELAYFSTON` `N... DELAYFSTOF`	命令单独在一个零件程序行中。**DELAY F**eed **ST**op **ON** /**OF**



说明

DELAYFSTON	定义一个域的开始，在这个范围中“软”停止被延后，直到到达停止延迟区的末尾。
DELAYFSTOF	定义一个停止延迟区的结尾
	两个指令都只允许在零件程序中，而不可以在同步动作中。
	在机床数据11550中：STOP_MODE_MASK Bit 0 = 0 (缺省)如果G331/G332有效，一个轨迹运动或G4被编程，则一个停止延迟区将被隐含定义。见下说明。



功能

在确定的程序段中不应被停止，进给也不应被改动。实质上短的程序段，例如用作螺纹生产的短程序段，会受到保护，使其几乎不产生停止。在程序段处理完后，一个可能的停止才能产生作用。
在停止延迟区中拒绝一个进给和进给禁止的改变。它们在停止延迟区之后才产生作用。

停止事件将被区分为：

- “软”停止事件　　反应：延迟
- “硬”停止事件　　反应：立即

事件名称	反应	注释
RESET	立即	VDI: DB21.DBB7.Bit7 和 DB11.DBB20.Bit7
PROG_END	报警16954	程序： M30
INTERRUPT	延迟	VDI: FC-9 和 DB10.DBB1 (Asup)
DELDISTOGO_SYNC	立即	VDI: 通道中和轴向删除剩余行程
PROGRESETREPEAT	延迟	VDI: DB21.DBB6.Bit3
PROGCANCELSUB	延迟	VDI: DB21.DBB6.Bit4
SINGLEBLOCKSTOP	延迟	VDI: 每个程序段之后停止 提示：接通停止延迟区中的单个程序的运行，NCK在停止延迟区外在第一个程序段的末尾停止。如果单个程序在停止延迟区前已被选择，NCK在每个程序界限上（也在停止延迟区中）停止。
SINGLEBLOCK_IPO	延迟	VDI: 开通单段类型－1
SINGLEBLOCK_DECODIER	延迟	VDI: 开通单段类型－2
STOPALL	立即	VDI: DB21.DBB7.Bit4 和 DB11.DBB20.Bit6
STOPPROG	延迟	VDI: DB21.DBB7.Bit3 和 DB11.DBB20.Bit5
OVERSTORE_BUFFER_END_REACHED	报警16954	NC程序：停止，因为清空溢出缓存器
PREP_STOP	报警16954	NC程序：STOPRE 和所有隐含的停止请求
PROG_STOP	报警16954	NC程序：M0 和 M1
STOPPROGATBLOCKEND	延迟	VDI: DB21.DBB7.Bit2
STOPPROGATASUPEND	系统故障	UP结束，撤消停止延迟区的选择。
WAITM	报警16954	NC程序： WAITM
WAITE	报警16954	NC程序： WAITE
INIT_SYNC	报警16954	NC程序： INIT带参数“S”
MMCCMD	报警16954	NC程序： MMC(STRING, CHAR)
PROGMODESLASHON	延迟	VDI: 程序段跳跃开或者转换
PROGMODESLASHOFF	延迟	VDI: DB21.DBB26 关闭程序段跳跃
PROGMODEDRYRUNON	延迟	VDI: DB21.DBB0.Bit6 开通空运行
PROGMODEDRYRUNOFF	延迟	VDI: DB21.DBB0.Bit6 关闭空运行
BLOCKREADINHIBIT_ON	延迟	VDI: DB21.DBB6.Bit1 关闭单段禁止
STOPATEND_ALARM	立即	警报：报警设计 STOPATENDBYALARM
STOP_ALARM	立即	警报：报警设计 STOPBYALARM
STOPATIPOBUFFER_ISEMPTY_ALARM	立即	内部：报警之后停止，清空插补缓存器
STOPATIPOBUF_EMPTY_ALARM_REORG	立即	内部：报警之后停止，清空插补缓存器
RETREAT_MOVE_THREAD	报警16954	NC程序：报警16954， 在LFON时（在停止&快速提刀时，在G33中不可以）
WAITMC	报警16954	NC程序： WAITMC
NEWCONF_PREP_STOP	报警16954	NC程序： NEWCONF
BLOCKSEARCHRUN_NEWCONF	报警16954	NC程序： NEWCONF
SET_USER_DATA	延迟	BTSS: PI "_N_SETUDT"
SYSTEM_SHUTDOWN	立即	在840Di时系统关闭
ESR	延迟	扩展的停止和退回
EXT_ZERO_POINT	延迟	外部零点偏移
STOPRUN	警报16955	BTSS: PI "_N_FINDST" STOPRUN

反应说明

立即 ("硬" 停止事件) 立即在停止延迟区中停止

延迟 ("软" 停止事件) 停止（也包括短时间的）只在停止延迟区后产生。

警报16954 程序将被停止，因为在停止延迟区中使用了不被允许的程序命令。

警报16955 在停止延迟区中进行一个不被允许的动作，程序继续。

停止延迟区的优点：

在没有速度干扰的情况下，处理程序段。

用户在停止后使用复位中断程序，这就使中止的程序段在被保护的范围之后。这样的程序段适用于作为一个跟踪搜索的搜索目标。

只要一个停止延迟区被加工，则以下的主运行轴不会被停止。

- 指令轴和
- 用POSA运行的定位轴

零件程序命令G4在停止延迟区中是被允许的，相反的其他的会导致暂时性的停止的零件程序命令不被允许。

G4如同一个轨迹运行一样，使停止延迟区有效，或者保持其有效性。

举例：进刀咬合

如果倍率在停止延迟区之前降低到6%，则该倍率在Stopp-Delay-（停止延迟区）有效。

如果倍率在停止延迟区由100%降低到6%，则停止延迟区以100%运行到结束，然后以6%继续运行。

进刀禁止在离开停止延迟区停止后才产生影响。

叠加/嵌套：

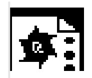
两个停止延迟区相交，一个来自语言指令，另一个来自机床数据11550，这样组成最大可能的停止延迟区。

以下几点作为规范，作为带嵌套和子程序结束的语言指令DELAYSTON和DELAYSTOF共同作用的说明：

1. 在DELAYFSTON被调用的子程序结束后，DELAYSTOF被隐含激活。
2. DELAYFSTON停止延迟区没有影响。
3. 如子程序1在停止延迟区中调用子程序2，那子程序2就是完整的停止延迟区。特别是DELAYSTOF在子程序2中无效。

附注：REPOSA是一个子程序，DELAYSTON在任何情况下将不再被选择。
如果一个“硬”的停止事件碰到“停止延迟区”，那“停止延迟区”完全不再被选择。也就是说如在这个程序段中又出现另一个任意的停止，则立即停止。
只有一个新的编程（新的DELAYFSTON)
才允许开始一个新的Stop-Delay-区。
如在停止延迟区前按停止键，并且NCK须
必须在停止延迟区中制动，则NCK在停止延迟区中停止，停止延迟区 不再被选择。
这适用于所有“软”停止事件。

用STOPALL可以在停止延迟区中制动。
用STOPALL使所有其它的停止事件立即生效，
它们一直延迟到现在。



编程举例

嵌套在两个程序级的停止延迟区：

```
N10010  DELAYFSTON()                    ；带N10xxx的程序级1的程序段
N10020  R1 = R1 + 1
N10030  G4 F1                           ；开始停止延迟区
...
N10040  子程序2
...
        ...                             ；子程序2的插补
        N20010  DELAYFSTON()            ；程序级2重复的开始，无效
        ...
        N20020  DELAYFSTOF()            ；另一个程序级的结束，无效
        N20030  RET

N10050  DELAYFSTOF()                    ；相同级的停止延迟区的结尾。
...
N10060  R2 = R2 + 2
N10070  G4 F1                           ；结束停止延迟区停止从现在起直接生效
```

系统变量：
一个停止延迟区可以用**$P_DELAYFST**在零件程序中鉴别。如果系统变量中的位0值为1的话，零件程序的加工在这个时候处于一个停止延迟区内。
一个停止延迟区可以用**$P_DELAYFST**在同步动作中鉴别。如果系统变量中的位0值为1的话，零件程序的加工在这个时候处于一个停止延迟区内。



其它说明

兼容性：
机床数据 11550:STOP_MODE_MASK 中的预置 位 0 = 0 隐含作用停止延迟区，
当有G指令G331/G332，并且已经编程了一个轨迹运行或者G4。

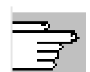
当有G指令G331/G332，并且已经编程了一个轨迹运行或者G4时，位0＝1使产生停止。
为了定义一个停止延迟区，必须使用指令DELAYFSTON/DELAYFSTOF 。

编程举例摘录

；在一个循环中重复以下的程序段

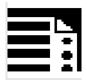
```
...
N99   MY_LOOP:
N100  G0 Z200
N200  G0 X0 Z200
N300  DELAYFSTON()
N400  G33 Z5 K2  M3 S1000
N500  G33 Z0 X5 K3
N600  G0 X100
N700  DELAYFSTOF()
N800  GOTOB  MY_LOOP
...
```


在图中可以看出用户已经在停止延迟区按下“停止”，并且NC在停止延迟区之外开始制动过程，比如在程序段N100中。因此NC在前面的N100区中进入停止。



有关类型SERUPRO的程序段搜索的详细情况参见
/FB/, K1 , 通道, 程序运行，剩余性能。
/FB/, V1,*进给，与* G331/G332
一起进给，攻丝不带补偿轴衬。

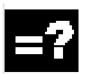
9.8 阻止程序点，用于 SERUPRO, IPTRLOCK, IPTRUNLOCK



编程

```
N... IPTRLOCK
N... IPTRUNLOCK
```

这些指令单独存在在一个零件程序行中，并且提供一个可编程的中断指示



说明

IPTRLOCK	开始不可查找的程序段
IPTRUNLOCK	结束可查找的程序段
	两个指令都只允许在零件程序中，而不可以在同步动作中



功能

对于某些机械复杂的机床配置，要求阻止程序段查找SERUPRO。使用一个可编程的中断指示，可以使“查找中断点”时在不可查找的位置之前停止。也可以在零件程序范围中定义不可查找的区域，在其中NCK不可以再次进入。使用程序中断，NCK记下最后加工的程序段，通过操作界面HMI可以查找到该程序段。



工作流程

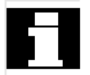
采集和查找不可查找的区域

不可查找的程序段用语言指令IPTRLOCK和IPRTUNLOCK标识。

指令IPTRLOCK使中断指针停止在主运行中可执行的单段(SBL1)上。该程序段在下面作为停止程序段标识。如果在IPTRLOCK之后出现一个程序中断，则在操作界面HMI上可以查找这个所谓的停止程序段。

再次停止在当前的程序段

用IPTRUNLOCK使中断指针在后面的程序段中停止在产生中断点的当前程序段。



在找到一个查找目标后，可以用该停止程序段重复一个新的查询目标。

一个由用户编辑的中断指针必须通过HMI再次去除。



其它说明

嵌套规则：

以下几点作为规范，作为带嵌套和子程序结束的语言指令IPTRLOCK 和 IPTRUNLOCK共同作用的说明：

1. 随着调用IPTRLOCK的子程序结束，IPTRUNLOCK隐含激活。
2. IPTRLOCK 在一个不可查找的区域没有作用。
3. 如果子程序1在一个不可查找的区域调用子程序2，则子程序2保持不可查找。特别是IPTRUNLOCK 在子程序2中无效。

系统变量：

一个无法搜索到的区域可以用**$P_IPRTLOCK**在零件程序中识别。

编程举例

两个程序级中的、带隐含的IPTRUNLOCK的，无法搜索到的程序段的嵌叠。这个隐含的子程序1中的IPTRUNLOCK结束这个无法搜索到的域。

N10010 IPTRLOCK()	
N10020 R1 = R1 + 1	
N10030 G4 F1	；停止程序段
...	；开始不可查找的程序段
N10040 子程序2	
...	；子程序2的说明
N20010 IPTRLOCK ()	；重复的开始，无效
...	
N20020 IPTRUNLOCK ()	；另一个程序级的结束，无效
N20030 RET	
...	
N10060 R2 = R2 + 2	
N10070 RET	；不能搜索到的程序段的结束
N100 G4 F2	；继续主程序

中断到100，重新提供了中断指示。

自动的中断指示

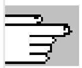
自动的中断指示的功能自动将一个先前确定的耦合方式确定为无法搜索。使用机床数据

MD 22680:AUTO_IPR_LOCK 可以用于

- 电子齿轮（在EGON时）
- 轴向引导值耦合（在LEADON时）

激活自动中断指针。

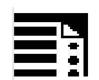
如果编程的和自动的中断指示（一个从语言指令中来，其他的从MD22680中来：AUTO_IPR_LOCK）相交在一起，那么将构成最大可能的不可查找的区域。



其它更多信息参见
/FB/，K1，通道，程序运行，剩余特性。



9.9 再次返回运行到轮廓, REPOSA/L REPOSQ/H, RMI/N RMB/E



编程

REPOSA RMI DISPR=… **或者** REPOSA RMB **或者** REPOSA RME **或者** REPOSA RMN

REPOSL RMI DISPR=… **或者** REPOSL RMB **或者** REPOSL RME **或者** REPOSL RMN

REPOSQ RMI DISPR=… DISR=… **或者** REPOSQ RMB DISR=… **或者** REPOSQ RME DISR=… **或者** REPOSQA DISR=…

REPOSH RMI DISPR=… DISR=… **或者** REPOSH RMB DISR=… **或者** REPOSH RME DISR=… **或者** REPOSHA DISR=…



指令说明

返回行程

REPOSA	所有轴线性返回
REPOSL	以线性返回
REPOSQ DISR=…	以四分之一圆弧返回，半径DISR
REPOSQA DISR=…	所有轴以四分之一圆弧返回，半径DISR
REPOSH DISR=…	以半个圆弧返回，直径DISR
REPOSHA DISR=…	所有轴以半个圆弧返回，半径DISR

再次返回点

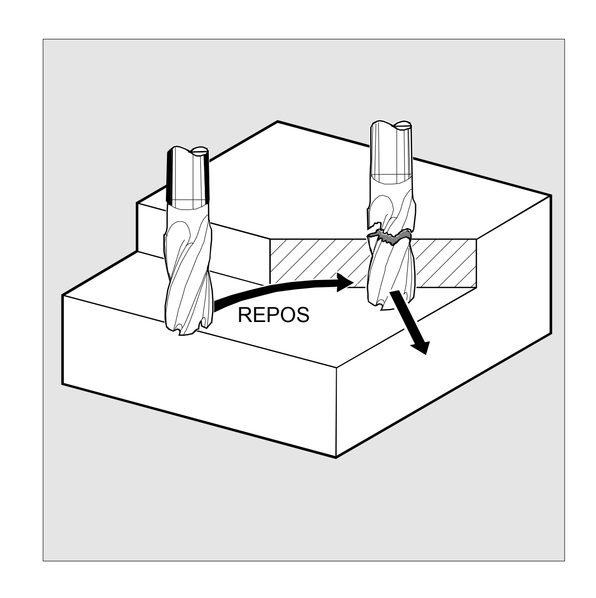
RMI	返回中断点
RMI DISPR=…	在中断点之前的进入点，间距DISPR，单位毫米/英寸
RMB	返回程序段起始点
RME	返回程序段终点
RME DISPR=…	返回程序段终点，间距DISPR，在终点之前
RMN	返回到下一个轨迹点
A0 B0 C0	应该返回的轴

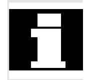


功能

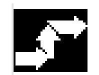
如果您在加工过程中必须中断正在运行的程序，使刀具移开，比如由于刀具断裂或者需要测量尺寸，则您可以通过程序控制再次返回轮廓到一个可选择的点。

REPOS指令作用如同一个子程序跳转（比如通过M17）。在中断程序中的后续程序段不再执行。





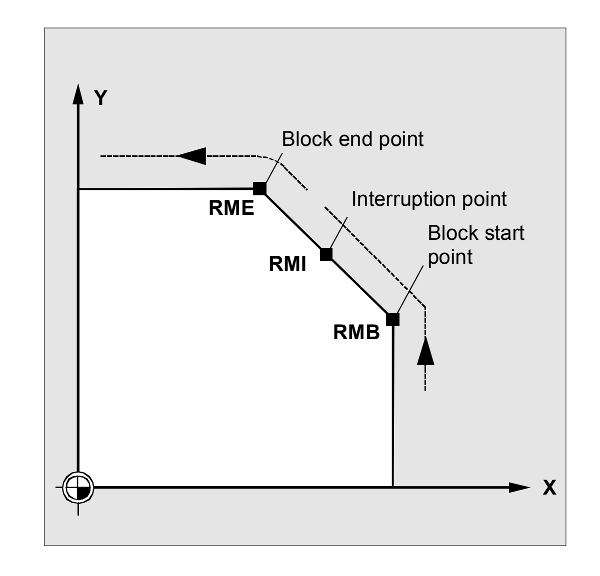
对于程序过程的中断，参见编程说明“工作准备”中的“中断程序”段落。



工作流程

确定再次返回点(至 SW 6.2)
考虑到程序中断的NC程序段，您可以在三个再次返回点之间选择：

- RMI，中断点
- RMB，程序段起始点或者最后的终点
- RME，程序段终点



用RMI DISPR=… 或者用RME DISPR=…
可以确定一个再次返回点，这个点在中断点之前或者在程序段终点之前。
用DISPR=…可以以毫米/英寸来描述轮廓轨迹，再次返回点在中断点或者终点**之前**。这个点最大可以位于程序段起始点。
如果没有编程DISPR=…，那么DISPR=0有效，并且在此中断点（在RMI时）或者程序段起始点（在RME时）有效。

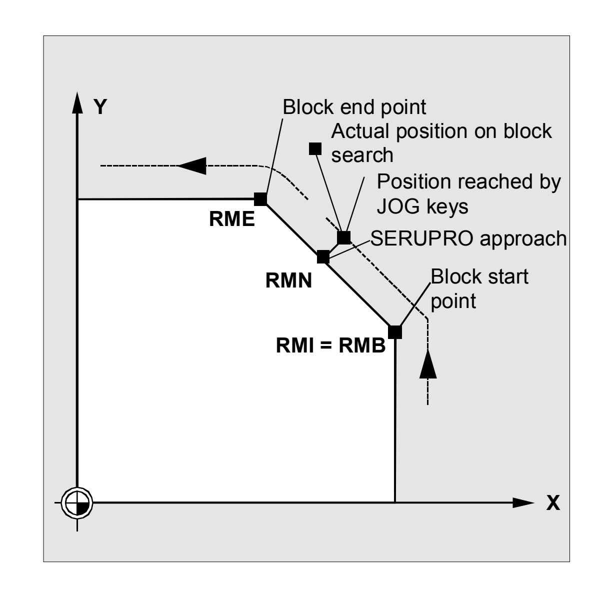
自软件版本SW5.2起：
计算DISPR前面的符号。在正号时，特性同前。
负号时，在中断点之后或者在起始点之后的RMB时会再次搁置。
中断点－搁置点之间的距离由DISPR的量产生。对于总量较大的值，该点最大可以位于程序段终点处。

应用示例：

通过一个传感器可以识别出到夹板的接近。释放一个ASUP，用此可以绕行该夹板。接着使用负DISPR再次定位到夹板之后的一个点，并且继续该程序。

用RMN返回SERUPRO（自软件版本SW 6.3起）

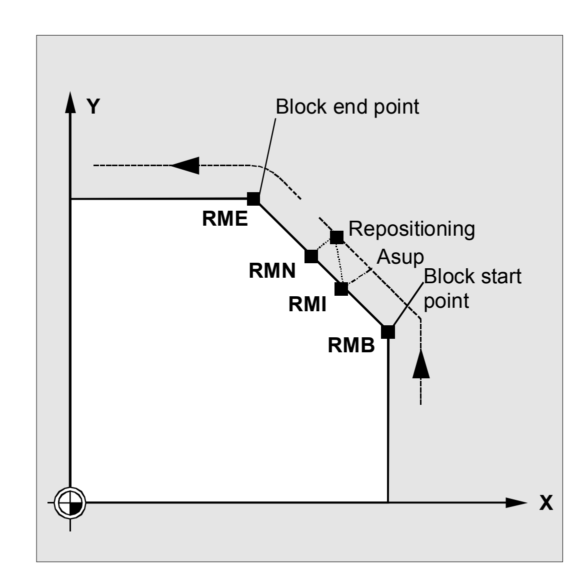
如果在加工过程中，必须在一个任意位置停止，则用SERUPRO，在RMN下以到中断点最短的位移返回运行，以便接着仅加工剩余行程。对此用户启动一个SERUPRO过程到中断程序段，并用JOG键定位到目标程序段损坏位置之前。



对于SERUPRO来说RMI和RMB是一样的。RMN不是仅仅限制到SERUPRO，而是共同有效。

返回运行最近的轨迹点RMN

在说明REPOSA的时刻，在中断之后，带RMN的再次返回程序段不再完整地再次开始，而是仅仅执行剩余行程。返回运行到中断程序段的最近轨迹点。



自软件版本SW 6.4起 系统变量：

中断程序段有效的 REPOS方式可以通过变量

$AC_ REPOS_PATH_MODE 由同步动作读出:

0: 不定义返回运行

1 RMB: 返回到开始

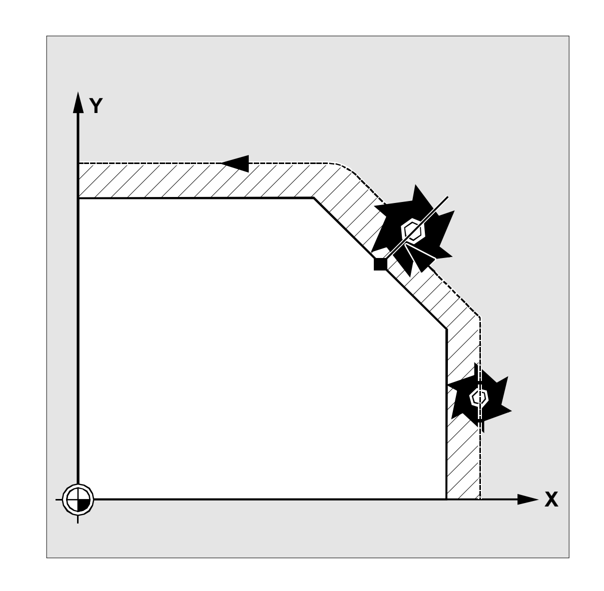
2 RMI: 返回到中断点

3 RME: 返回到程序结束点

4 RMN: 返回到中断程序段
最近的轨迹点。

以新刀具返回运行

如果程序运行由于刀具损坏而停止：
通过编程新的D号，该程序自再次返回点起以修改后的刀具补偿值继续进行。





如果刀具补偿值修改，则中断点可能不再返回。在这种情况下，返回到新轮廓上与该中断点最近的点（有时更改DISPR）。

返回轮廓

刀具的这种再次返回轮廓的运动可以编程。用值零说明待运行轴的地址。

使用指令 REPOSA, REPOSQA 和 REPOSHA，
所有轴自动重新定位。不需要进行轴说明。

在编程REPOSL、REPOSQ和REPOSH时，所有几何轴自动返回运行，即使在指令中没有说明。所有其它待重新定位的轴必须在指令中说明。

以直线返回, REPOSA, REPOSL

刀具以一条直线直接回到再次返回点。

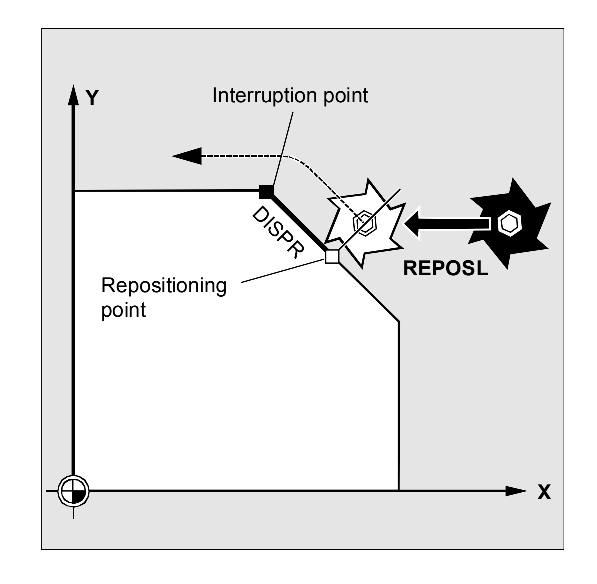
使用REPOSA，所有轴自动处理。使用REPOSL时，您可以指定待运行的轴。

举例：

```
REPOSL RMI DISPR=6 F400
```

或者

```
REPOSA RMI DISPR=6 F400
```

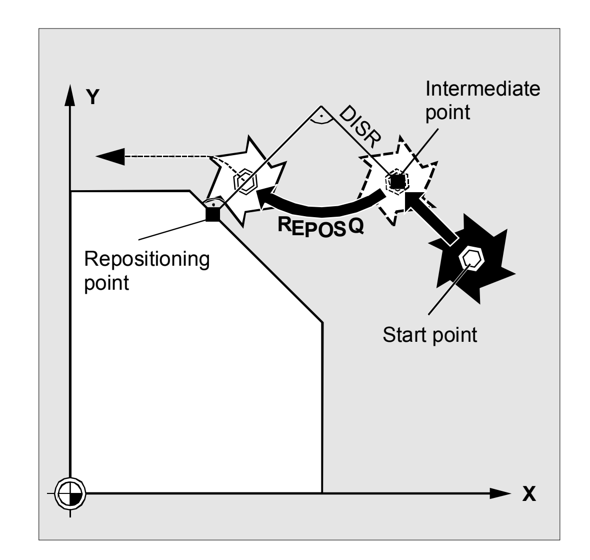


以四分之一圆弧返回, REPOSQ, REPOSQA

刀具以四分之一圆弧，半径DISR=...返回运行到再次返回点。控制系统自动计算起始点和再次返回点之间所必需的中间点。

举例：

```
REPOSQ RMI DISR=10 F400
```

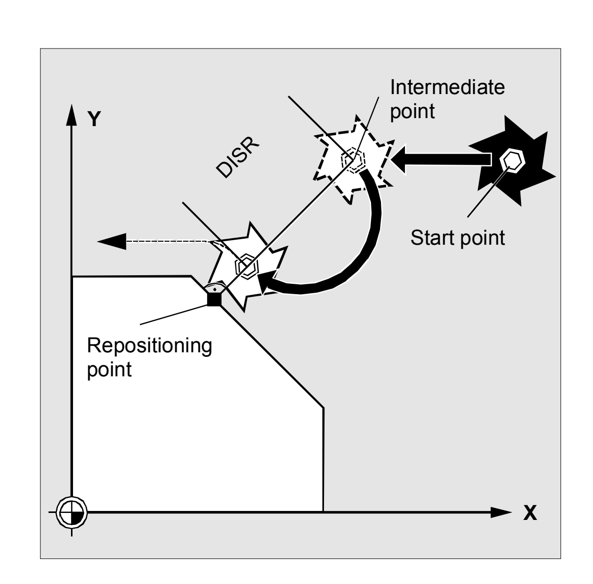


以半圆返回, REPOSH, REPOSHA

刀具以半圆，直径DISR=...返回运行到再次返回点。控制系统自动计算起始点和再次返回点之间所必需的中间点。

举例：

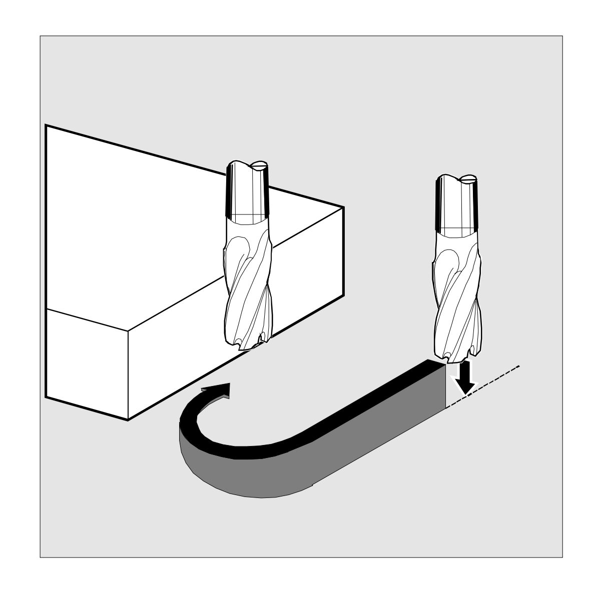
```
REPOSH RMI DISR=20 F400
```



REPOSH 和 REPOSQ适用于圆弧运动：

圆弧在所说明的工作平面G17到G19中运行。

如果在返回运行程序段中您指定了第三个几何轴（横向进给方向），并且刀具位置和编程的位置在横向进给方向不一致，则以一条螺旋线返回运行到再次返回点。



在下面的情况下自动转换到线性返回运行REPOSL：

- 您没有指定DISR的值。
- 没有定义的返回运行方向（在一个程序段中程序中断，没有运行信息）。
- 当返回运行方向垂直于当前工作平面时。

■

用于记录

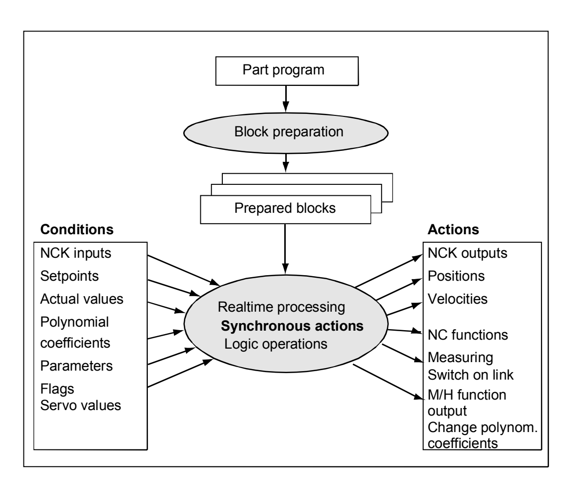
灵活的NC编程

10.1 结构, 基础部分



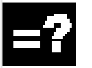
功能

同步动作提供如下可能性：从当前的零件程序出发推动几个不同的动作，并使它们同步执行。
同步动作如何使用由条件定义，其求值运算以实时（插补节拍）方式进行。这些动作是对实时事件的反应；执行并不是在程序段交接处进行。
此外，同步动作还包含动作有效级的说明和对编程实时变量的询问频率，以及对启动动作的执行频率说明。由此，一个动作可以一次或者也可以循环（插补节拍）方式进行触发。



编程

DO 动作1 动作2 …
关键字条件 DO 动作1 动作2 …
ID=n 关键字条件 DO 动作1 动作2 …
IDS=n 关键字条件 DO 动作1 动作2 …



说明

识别号 ID/IDS	
ID=n	**模态** 有效同步动作，自动方式运行, **程序局部**; n = 1... 255
IDS=n	**模态** 有效同步动作，在每种运行方式, **静态**; n = 1... 255
ohne ID/IDS	**程序段方式** 有效同步动作，自动方式运行
关键字	
没有关键字	动作执行不受条件制约。在每个插补节拍循环执行动作。

WHEN	一直检查该条件，直至满足；相关的动作执行一次。
WHENEVER	循环检查该条件。该条件一旦满足，则循环执行相关动作。
FROM	条件一次满足以后，循环执行该动作，只要同步动作有效。
EVERY	条件满足后，动作触发一次，并且当状态从FALSE转换到TRUE时再执行一次。循环检查该条件。每次满足条件均执行一次相关的动作。
条件	实时变量的连接逻辑，在IPO节拍中检查该条件。 自软件版本5起，用于计算条件的同步动作的G代码可以编程。
DO	在条件满足后释放动作。
Aktion	条件满足后启动动作，比如变量赋值，接通轴耦合，设置NCK输出端，输出M-,H-功能，... 自软件版本5起，用于动作/工艺循环的同步动作的G代码可以编程。
协调同步动作/工艺循环	
CANCEL[n]	删除同步动作
LOCK[n]	禁止工艺循环
UNLOCK[n]	释放工艺循环
RESET	复位工艺循环

编程举例

WHEN $AA_IW[Q1]>5 DO M172 H510	；如果轴Q1的实际值超过5毫米，则辅助功能M172和H510输出到PLC接口。

如果在一个零件程序中出现实时变量（比如实际值，一个数字输入端或者输出端的排列等等），则停止进刀，直至前一个程序段执行完毕，并且出现实时变量值。
所使用的实时变量在插补节拍中求算。

在同步动作时的优点：

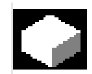
这里没有出现进刀停止。

可能的应用：

- 优化运行时间紧张的应用场合
  （比如换刀）

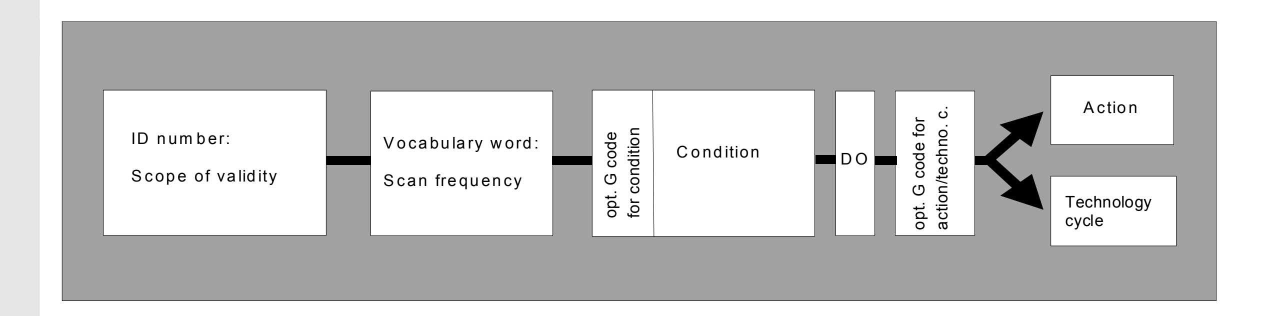
- 对外部事件的快速反应
- 编程AC调节
- 调节安全功能
- ....

10.1.1 编程和指令单元



功能

在程序段中仅有一个同步动作，并且在一个机床功能
的下一个可执行的程序段中起作用
（比如用G0,G1,G2,G3的运行）。
同步动作由最多5个指令单元
组成，带有不同的任务：



举例:

ID=1	WHENEVER	$A_IN[1]==1	DO	$A_OUT[1]=1
同步动作号1:	如果	有输入端1	则	设置输出端1

10.1.2 有效性范围：识别号 ID



功能

一个同步动作的有效性范围通过识别号（模态ID）确定：

- 没有模态ID
  同步动作仅在自动方式运行时生效。它仅适用于后面可执行的程序段（程序段带运行指令或者其它机床动作），也是程序段方式生效。
  举例：

```
WHEN $A_IN[3]==TRUE DO $A_OUTA[4]=10
G1 X20                                    ；可执行的程序段
```

- **ID**=n; n=1...255
  同步动作在后面的程序段中模态有效，通过CANCEL(n)或者通过编程一个新的、具有相同ID的同步动作取消。
  在M30程序段中有效的同步动作继续有效（在有些情况下用CANCEL指令删除）。
  ID同步动作仅在自动方式运行时生效。
  举例：

```
ID=2 EVERY $A_IN[1]==1 DO POS[X]=0
```

- **IDS**=n; n=1...255
  静态同步动作在所有运行方式模态有效。
  它不仅可以在零件程序中定义，而且也可以在上电之后直接从一个由PLC启动的异步子程序（ASUP）中定义。因此可以激活动作，它们与NC中所选择的运行方式无关，直接运行。
  举例：

```
IDS=1 EVERY $A_IN[1]==1 DO POS[X]=100
```



应用：

- 在JOG方式下的AC循环
- 用于安全集成的连接逻辑
- 监控功能，对所有运行方式中机床状态的反应

加工顺序

模态和静态有效的同步动作按照其ID(S)序号（插补节拍）的顺序进行加工。
程序段方式有效的同步动作（没有ID号）在加工模态有效的同步动作结束之后，按照编程的顺序进行处理。

10.1.3 关键字



功能

该关键字确定如何经常询问下面的条件，以及如何经常执行相关的动作：

- 没有关键字：
  如果没有编程关键字，则该条件一直被看作满足。
  同步指令循环执行。

举例：

```
DO $A_OUTA[1]=$AA_IN[X]
        ；实际值输出到模拟量输出端
```

- WHEN
  该条件在每个插补节拍一直询问，直至满足一次，然后再精确执行一次该动作。
- WHENEVER
  该条件在每个插补节拍询问。该条件一旦满足，则在每个插补节拍执行该动作。
- FROM
  在每个插补节拍检查该条件，直至满足一次。只要同步动作激活，该动作一直执行，也就是说在该条件不再满足后也如此。
- EVERY
  在每个插补节拍询问该条件。该条件一旦满足，则始终执行该动作一次。
  脉冲沿控制：
  如果该条件从状态FALSE转换到TRUE，则再次执行该动作。

举例：

```
ID=1 EVERY $AA_IM[B]>75 DO
POS[U]=IC(10) FA[U]=900;
```

如果在MKS中的轴B实际值超出75，则U轴在轴向进给方向移动10定位。

条件

比较两个实时变量，或者比较一个实时变量与前面计算的表达式，确定是否应该执行一个动作。

自软件版本SW4起：

在条件中，比较结果也可以通过布尔运算符进行逻辑联系（）。

在插补节拍中检查该条件。如果该条件满足，则执行相关的动作。

自软件版本SW5起：

条件可以用一个G代码说明。这样可以实现：与刚刚有效的零件程序状态无关，形成经过定义的状态，用于求算条件以及待运行的动作/工艺循环。要求同步动作去除与程序外围的耦合，因为在任意时间根据所满足的释放条件，在定义输出状态执行同步动作。

应用情况：

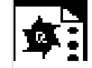
确定用于条件求值和动作的尺寸系统，使用G代码

G70, G71, G700, G710。

在软件版本5中仅允许这些G代码。

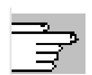
如果在该动作中没有说明自身的G代码，则在条件中所说明的G代码适用于求算条件和动作。

每个条件部分仅允许编程G代码组中的一个G代码。



编程举例

`WHENEVER $AA_IM[X] > 10.5*SIN(45) DO …`	与进刀运行中计算的表达式进行比较
`WHENEVER $AA_IM[X] > $AA_IM[X1] DO …`	与接下去的实时变量进行比较
`WHENEVER ($A_IN[1]==1) OR ($A_IN[3]==0) DO ...`	两个相互逻辑联系的比较



可能的条件：

- 比较实时变量（模拟/数字输入/输出，等等）
- 两个比较结果之间的布尔逻辑联系
- 计算实时表达式
- 时间/距离程序段开始
- 距离程序段结束
- 测量值，测量结果
- 伺服值
- 速度，轴状态



10.1.4 动作

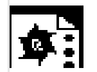


功能

在每个同步动作中可以编程一个或者多个动作。所有在一个程序段中编程的动作以相同的插补节拍启动。
自软件版本5起，带一个G代码的动作可以用于动作/工艺循环。有时，在程序段中和工艺循环中所有的动作给定一个另外的G代码，与在条件中所设置的不同。如果在动作部分有工艺循环，则该G代码在工艺循环结束之后，也适用于所有后续的动作，直至下一个G代码模态有效。
每个动作部分仅允许编程G代码组（G70, G71, G700, G710）中的一个G代码。

可能的动作：

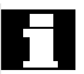
- 变量赋值
- 写设定数据
- 设置系统参数
- DELDTG:快速删除剩余行程
- RDISABLE:设置读入禁止
- 输出M-,S-和H辅助功能
- STOPREOF:取消进刀停止
- FTOC:在线刀具补偿
- 定义计算功能（多项式）
- SYNFCT:激活计算功能：AC调节
- 在一个编程的程序段中在几个进给之间转换，取决于二进制或者模拟量信号
- 进给补偿
- 定位轴（POS）和主轴（SPOS）启动/定位/停止
- PRESETON:设定实际值
- 激活或者取消联动/引导值耦合
- 测量
- 调整附加的安全功能
- 输出数字和模拟量信号
- ...



编程示例:

同步动作，带两个动作

```
WHEN $AA_IM[Y] >= 35.7 DO M135 $AC_PARAM=50
        ；如果条件满足，则M135输出到PLC，并且倍率设置到50%
```



一个程序（单轴程序，工艺循环）也可以作为动作说明。
该程序由可以单独在同步动作中编程的动作构成。这样一个程序中的各个动作按插补节拍中顺序执行。



说明

动作可以与运行方式无关执行。
在有效程序中，仅在自动方式下以下动作生效

- STOPREOF,
- DELDTG.

10.1.5 同步动作一览

至软件版本SW3.x

- 在用户级面（零件程序）编程插补节拍顺序
- 对插补节拍中事件/状态的应答
- 实时状态逻辑联系
- 对外设、控制系统状态和机床状态进行存取
- 编程循环过程顺序，它们以插补节拍执行
- 释放专门的NC功能（读入禁止，轴向叠加运动，…）
- 加工工艺功能，与轨迹运行并行
- 释放工艺功能，与程序段界限无关

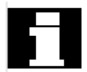
自软件版本SW4起：

- 用于同步动作的诊断方法
- 扩展在同步动作中可使用的主运行变量
- 同步动作的综合条件
- 扩展同步动作表达式：
  逻辑联系实时变量与基本运算方式和插补节拍功能，
  间接编址主运行变量，带变址，可在线修改；同步动作设定数据，可在线修改和求算
- 可设计性：通过机床数据可以设定同时有效同步动作的个数。
- 定位轴运动，从同步动作启动主轴（指令轴）
- 由同步动作预设定
- 开/关，设定轴耦合参数：引导值耦合，联动
- 开/关轴向测量功能
- 软件凸轮
- 删除剩余行程，没有进刀停止
- 单个轴程序，工艺循环
- 在JOG方式下同步动作生效，跨过程序界限
- 同步动作可以由PLC影响
- 保护的同步动作
- 扩展用于叠加运动/距离调整

自软件版本SW5.x

- 碰撞到固定挡块FXS：
  用FXS, FXST 和 FXSW释放同步动作
- 以限制的力矩/力FOC运行：
  同步动作模态或者按程序段用FOCON激活，或者用FOCOF取消

10.2 用于条件和动作的基本模块



实时变量

实时变量 以插补节拍求算和写入。

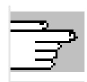
实时变量是

- $A… , 主运行变量，
- $V… , 伺服变量。

为了进行特殊标记，可以把同步动作中的这些变量以**$$**编程：

$AA_IM[X] 与 $$AA_IM[X] 数值相同。

如果求值/赋值以插补节拍进行，则设定数据和机床数据必须用$$标记。



变量清单参见附录。



实时计算

实时计算限制为数据类型INT, REAL 和 BOOL。

实时表达式是插补节拍中可执行的计算，它们可以在条件和动作中用于赋值NC地址和变量。

- 比较
  在条件中可以比较相同数据类型的变量或者部分表达式。结果始终为数据类型BOOL。
  可以使用所有已知的比较运算符
  (==, <>, <, >, <=, >=)。
- 布尔运算符
  变量、常量或者比较符可以与已知的布尔运算符相逻辑联系（NOT, AND, OR, XOR)。

- 位方式运算符
  可以有位方式运算符 B_NOT, B_AND, B_OR, B_XOR。
  运算数为变量或整数型常量。

- 基本运算方式
  整数型和实数型实时变量可以通过基本运算方式相互或者与常量逻辑联系 (+, -, *, /, DIV, MOD)。

- 数学功能
  数学功能可以运用于实数型实时变量 (SIN, COS, TAN, ASIN, ACOS, ABS, TRUNC, ROUND, LN, EXP, ATAN2, POT, SQRT, CTAB, CTABINV)。
  举例:

```
DO $AC_PARAM[3] = COS($AC_PARAM[1])
```



说明：

仅可以是相同数据类型的变量之间进行逻辑联系。

正确：　$R10=$AC_PARAM[1]

错误：　$R10=$AC_MARKER[1]

先乘除后加减，表达式允许使用括号。

运算符DIV和MOD也允许用于实数型数据（自软件版本SW4起）。

举例：

```
DO $AC_PARAM[3] = $A_INA[1]-$AA_IM[Z1]          ；两个实时变量相减
WHENEVER $AA_IM[x2] < $AA_IM[x1]-1.9 DO $A_OUT[5] = 1
                                               ；从实时变量中减去一个常量
DO $AC_PARAM[3] = $INA[1]-4*SIN(45.7 $P_EP[Y])*R4
                                               ；常量表达式，在进刀时计算
```

- 变址

实时变量可以用实时变量变址。



说明：

不是实时方式构成的变量不允许用实时变量变址。

举例：

```
WHEN…DO $AC_PARAM[$AC_MARKER[1]] = 3
```

不允许：

```
$AC_PARAM[1] = $P_EP[$AC_MARKER]
```



编程举例

实时表达式举例

ID=1 WHENEVER ($AA_IM[Y]>30) AND ($AA_IM[Y]<40) DO $AA_OVR[S1]=80	选择一个位置窗口
ID=67 DO $A_OUT[1]=$A_IN[2] XOR $AN_MARKER[1]	求算2个布尔信号
ID=89 DO $A_OUT[4]=$A_IN[1] OR ($AA_IM[Y]>10)	输出一个比较结果



10.3 用于同步动作的专门实时变量

在同步动作中允许使用以下的实时变量：

10.3.1 标志位/计数器 $AC_MARKER[n]



功能

标志位变量可以读入/写入到同步动作。

通道专用的标志位/计数器 $AC_MARKER[n]

数据类型：INTEGER

在相同的名称下，每个通道只可以出现一个通道专用的标志位变量。

举例：

```
WHEN ... DO $AC_MARKER[0] = 2
WHEN ... DO $AC_MARKER[0] = 3
WHEN $AC_MARKER == 3 DO $AC_OVR=50
```

10.3.2 定时器变量 $AC_TIMER[n], 自软件版本 SW 4起



功能
(不包括 840D NCU 571, FM-NC)

系统变量$AC_TIMER[n]
可以在定义的等待时间之后启动动作。
数据类型：REAL
单位： s
n:定时器变量序号

- **设置定时器**
  通过赋值
  $AC_TIMER[n]=值，向上计数定时器变量
  n:时间变量序号 值：起始值（等同于0）

- 停止定时器
  定时器变量停止向上计数，通过赋值一个负值
  $AC_TIMER[n]=-1
- 读出定时器
  可以在运行或者停止的定时器变量中读出实际时间
  值。.通过赋值-1停止定时器变量，
  最后有效的时间值保持不变，并可以被继续读出。

举例:
在识别出一个数字输入端后，通过模拟量输出端输出一个
实际值500ms。

```
WHEN $A_IN[1]==1 DO $AC_TIMER[1]=0            ；复位定时器并启动
WHEN $AC_TIMER[1]>=0.5 DO $A_OUTA[3]=$AA_IM[X] $AC_TIMER[1]=-1
```

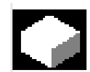
10.3.3 同步动作参数$AC_PARAM[n]



功能

数据类型： REAL
n:参数0-n序号
同步动作参数$AC_PARAM[n]用于计算，并作为同步
动作的中间存储器。
每个通道可使用AC参数变量的个数通过机床数据
MD28254:MM_NUM_AC_PARAM确定。
每个通道在相同的名称下参数只可以出现一次。标志
位$AC_PARAM保持在动态存储器中。

10.3.4 存取R参数$Rxx



功能

数据类型： REAL

这些静态变量用于零件程序中的计算。这些变量可以在插补节拍中通过添加**$**进行存取。

举例：

`WHEN $AA_IM[X]>=40.5 DO $R10=$AA_MM[Y]`	对R参数10进行读存取。
`WHEN $AA_IM[X]>=6.7 DO $R[$AC_MARKER[1]]=30.6`	；对R参数进行读存取，其序号位于标志位1中。



说明：

应用：

同步动作中R参数的应用，可以提供：

- 存储数值，它们在程序结束、NC复位和上电时保持不变。
- 在R参数图中显示所存储的值
- 存档同步动作时所求得的值。

R参数可以作为“通常”的计算变量Rxx使用，也可以作为实时变量$Rxx使用。

如果R参数在一个同步动作中使用之后，应该可以作为“通常”的计算变量使用，则明确编程带STOPRE的进刀停止必须用于进刀运行和主运行的同步。

举例：

`WHEN $AA_IM[X]>=40.5 DO $R10=$AA_MM[Y]`	在同步动作中使用R10
`G01 X500 Y70 F1000`	
`STOPRE`	进刀停止
`IF R10>20`	求算计算变量

10.3.5 读写机床数据和设定数据, 自软件版本 SW 4起



功能

自软件版本SW4起，可以从同步动作读写机床数据和设定数据（MD，SD）。

- **读不可改变的MD，SD**
  它们从同步动作中编址，如同通常的零件程序指令，并且用一个**$**符号导入。
  举例:

```
ID=2 WHENEVER $AA_IM[z]<$SA_OSCILL_REVERSE_POS2[Z]-6 DO $AA_OVR[X]=0
    ；作为不可改变的反向区域2触发用于摆动。
```

- **读可变的MD,SD**
  从同步动作中编址带 **$$-**
  符号的数据，并以插补节拍求算。
  举例:

```
ID=1 WHENEVER $AA_IM[z]<$$SA_OSCILL_REVERSE_POS2[Z]-6 DO $AA_OVR[X]=0
    ；这里从下面情况出发：在加工期间反向位置可以通过操作进行改变。
```

- **写MD, SD**
  前提条件：
  当前设定的存取权必须允许写存取。只有修改立即生效，从同步动作中修改MD和SD才有意义。用于所有MD和SD的有效级说明，参见： /LIS/, 清单
  编址：
  待修改的MD和SD用**$$**导入，并编写地址。
  举例:

```
ID=1 WHEN $AA_IW[X]>10 DO $$SN_SW_CAM_PLUS_POS_TAB_1[0]=20
                          $$SN_SW_CAM_MINUS_POS_TAB_1[0]=30
          ；改变SW凸轮的开关位置。说明：开关位置必须在到达位置之前修改2-3个插补节拍。
```



10.3.6 FIFO-变量 $AC_FIFO1[n] … $AC_FIFO10[n], 自软件版本 SW 4起



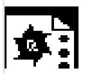
功能

数据类型： REAL
有10个FIFO变量（循环存储器）用于存储相关联的数据序列。
应用：

- 循环测量
- 循环加工

可以对每个单元进行读写存取。
可以使用的FIFO变量的个数可以通过机床数据 MD 28260:NUM_AC_FIFO确定。
在一个FIFO变量中可记入的值的个数通过机床数据 MD 28264:LEN_AC_FIFO定义。
所有FIFO变量有相同的长度。
只有当MD 28266:MODE_AC_FIFO Bit0 设置之后，才构成所有FIFO单元的和。

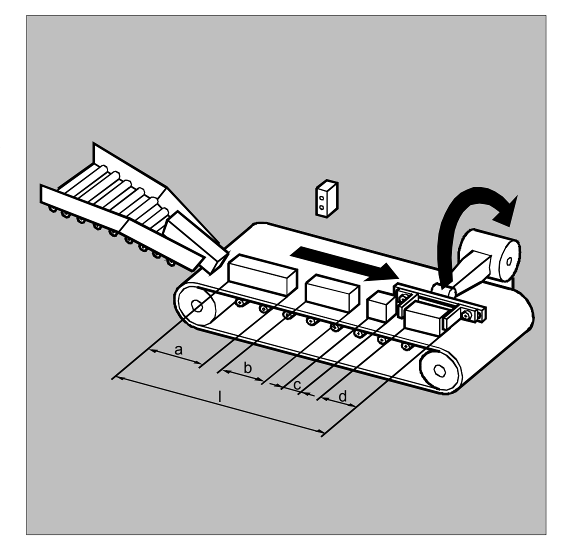
变址0到5有特殊的含义：
n=0: 在写入时：新值存放在FIFO中。
在读入时：读最早的单元并从FIFO中去除
n=1: 对最早存储的单元存取
n=2: 对最后存储的单元存取
n=3: 所有FIFO单元之和
n=4: FIFO中可以使用单元的个数。
可以对FIFO中的每个单元进行读写存取。
通过复位单元个数，比如第一个FIFO变量：
$AC_FIFO1[4]=0 复位FIFO变量。
n=5: 当前的写变址相对于FIFO起始
n=6 最大 $6+n_{max}$:
对第n个 FIFO-单元进行存取。



编程举例

循环存储器

在生产过程中，使用一个传送带用于传送不同长度(a, b, c, d)的产品。 因此在传送长度“I”的传送带上，同时传送不同数量的产品，这与其产品长度相关。如果传送速度不变，则传送带装卸产品处必须与产品的、不同到达时间相匹配。



`DEF REAL ZWI=2.5`	两个放置的产品之间恒定的间距。
`DEF REAL GESAMT=270`	在长度测量位置和装卸位置之间的距离。
`EVERY $A_IN[1]==1 DO $AC_FIFO1[4]=0`	在过程开始时复位FIFO。
`EVERY $A_IN[2]==1 DO $AC_TIMER[0]=0`	一个产品中断光栅时开始时间测量。

```
EVERY $A_IN[2]==0 DO $AC_FIFO1[0]=$AC_TIMER[0]*$AA_VACTM[B]
          ；如果光栅未遮挡，则从所测量的时间和传送速度计算产品长度，并存储到FIFO中。
EVERY $AC_FIFO1[3]+$AC_FIFO1[4]*ZWI>=GESAMT DO POS[Y]=-30
     $R1=$AC_FIFO1[0]
     ；一旦所有产品长度和中间间隔之和大于/等于产品放置位置和装卸位置之间的长度，则从传送带的
          装卸位置取下产品，并从FIFO中读出所属的产品长度。
```

10.3.7 有关插补器中程序段类型的信息 (SW 6.4 和 7.1)

下面的系统变量供同步动作使用，从而得到在主运行中当前程序段的信息：

$AC_BLOCKTYPE
$AC_BLOCKTYPEINFO
$AC_SPLITBLOCK

$AC_BLOCKTYPE 值：		$AC_BLOCKTYPEINFO 值：				
0 原始程序段	不等于0 中间程序段	T	H	Z	E	意义： 中间程序段触发器：
	1 (SW 6.4)	1	0	0	0	内部生成的程序段，没有其它信息 （自SW7.1）
(自SW 7.1起)		(自SW 7.1起)				
	2	2	0	0	1	倒棱/倒圆：直线
	2	2	0	0	2	倒棱/倒圆：圆弧
	3	3	0	0	1	WAB:以直线返回
	3	3	0	0	2	WAB:以1/4个圆弧返回
	3	3	0	0	3	WAB:以半圆返回
						刀具补偿：
	4	4	0	0	1	STOPRE之后的返回程序段
	4	4	0	0	2	在找不到交点时的连接程序段
	4	4	0	0	3	内角处（仅TRACYL）点状圆弧
	4	4	0	0	4	外角处绕行圆弧（或者锥形截面）
	4	4	0	0	5	去除补偿时返回程序段
	4	4	0	0	6	重新激活WRK时返回程序段
	4	4	0	0	7	由于曲率太大，分解程序段
	4	4	0	0	8	3D面铣削时补偿程序段 （刀具矢量II平面矢量）
						精磨削：
	5	5	0	0	1	G641
	5	5	0	0	2	G642
	5	5	0	0	3	G643
	5	5	0	0	4	G644

$AC_BLOCKTYPE 值：		$AC_BLOCKTYPEINFO 值：				
0	不等于0	T	H	Z	E	意义：
原始程序段	中间程序段					中间程序段触发器：
						TLIFT-程序段，带：
	6	6	0	0	1	切向轴线性运动，没有提刀运动
	6	6	0	0	2	切向轴非线性运动（多项式），没有提刀运动
	6	6	0	0	3	提刀运动，切向轴运动和提刀运动同时启动
	6	6	0	0	4	当到达确定的提刀位置时，首先启动提刀运动，切向轴
						位移划分：
	7	7	0	0	1	编程的位移划分有效，没有冲裁或者步冲
	7	7	0	0	2	编程的位移划分，带有效的冲裁或者步冲
	7	7	0	0	3	内部自动生成的位移划分
						编译循环：
	8	ID应用				编译－循环－应用的ID，它产生该程序段



说明：

如果一个中间程序段存在，则$AC_BLOCKTYPEINFO在千位（T）始终包含用于程序块类型的数值。($AC_BLOCKTYPE 不等于 0)

T	千位
H	百位
Z	十位
E	个位

在 $AC_BLOCKTYPEINFO中

$AC_SPLITBLOCK 值：	意义：
0	没有修改的编程的程序段，（通过压缩器生成的程序段也作为编程的程序段处理）。
1	有一个内部生成的程序段或者一个缩短的原始程序段。
3	内部生成的程序段或者缩短的原始程序段链中最后的程序段。



编程举例

计数精磨削程序段

```
$AC_MARKER[0]=0
$AC_MARKER[1]=0
$AC_MARKER[2]=0
...
；定义同步动作，以此计数精磨削程序段
；所有的精磨削程序段在$AC_MARKER[0]中计数
ID = 1 WHENEVER ($AC_TIMEC ==0) AND ($AC_BLOCKTYPE==5) DO _
      $AC_MARKER[0]= $AC_MARKER[0] + 1
...
；用G641产生的精磨削程序段在$AC_MARKER[1]中计数
ID = 2 WHENEVER ($AC_TIMEC ==0) AND ($AC_BLOCKTYPEINFO==5001) DO _
      $AC_MARKER[1]= $AC_MARKER[1] + 1
...
；用G642产生的精磨削程序段在$AC_MARKER[2]中计数
ID = 3 WHENEVER ($AC_TIMEC ==0) AND ($AC_BLOCKTYPEINFO==5002) DO _
      $AC_MARKER[2]= $AC_MARKER[2] + 1
...
```

10.4 同步进行的动作

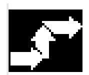
10.4.1 辅助功能输出



功能

在满足条件的情况下，每个加工程序段可以最多输出10个M-,H-和S-功能。
通过动作口令字 “DO” 触发辅助功能的输出。
以插补节拍立即输出辅助功能。通过机床数据定义的辅助功能输出时间无效。
当条件满足后，确定输出时间。

举例:
在某个轴位置时打开冷却液：
WHEN $AA_IM[X]>=15 DO M07 POS[X]=20 FA[X]=250



工作流程

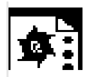
辅助功能仅允许用关键字WHEN或者EVERY，在程序段方式生效的同步动作中（没有模态ID）编程。辅助功能的有效性由PLC确定，比如通过NC启动。



说明：

从一个运动同步动作中不可以：

- M0, M1, M2, M17, M30:程序停止/结束（在工艺循环时可以为M2,M17,M30）。
- M70:主轴功能
- M6或者通过机床数据设定的M功能，用于刀具更换
- M40, M41, M42, M43, M44, M45:齿轮级切换



编程举例

`WHEN $AA_IW[Q1]>5 DO M172 H510`	如果Q1轴的实际值超过5毫米，则辅助功能M172和H510输出到PLC。



10.4.2 设置读入禁止RDISABLE



功能

使用 RDISABLE， 在条件满足时停止主程序中程序段的继续执行。编程的运动同步动作继续执行，后面的程序段继续处理。

在有RDISABLE的程序段的起始处，始终触发准停，而与RDISABLE是否有效无关。



编程举例

与外部的输入端无关，以插补节拍启动程序。

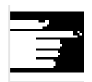
...	
WHENEVER $A_INA[2]<7000 DO RDISABLE	；如果输入端2低于7V的电压，则停止程序继续（1000＝1V）。
N10 G1 X10	；如果条件满足，则在N10结束处读入禁止生效
N20 G1 X10 Y20	
...	

10.4.3 取消进刀停止STOPREOF



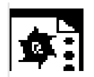
功能

一旦条件满足，则在明确编程的进刀停止STOPRE，或者一个有效的同步动作隐含有效的进刀停止时，STOPREOF在后一个加工程序段之后取消进刀停止。



说明：

STOPREOF必须用关键字WHEN以程序段有效方式（没有ID号）编程。



编程举例

在程序段结束处快速的程序分支。

WHEN $AC_DTEB<5 DO STOPREOF	；如果到程序段结束处的距离小于5毫米，则取消进刀停止
G01 X100	；在执行线性插补之后，取消进刀停止。
IF $A_INA[7]>500 GOTOF MARKE1=X100	；如果输入端7电压超过5V，则跳转到标签1



10.4.4 剩余行程删除

与一个条件相关，剩余行程删除可以释放，用于轨迹和所说明的轴。

可以使用：

- 快速、预备的剩余行程删除
- 删除剩余行程，没有准备（自软件版本SW4.3）

10.4.5 剩余行程删除，带预置，DELDTG, DELDTG(„轴 1 到 x“)



说明：

在DELDTG之后括号中的轴名称仅仅对一个定位轴有效。



功能

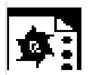
用DELDTG预置的剩余行程删除允许一个对释放事件的快速反应，因此在时间紧张的情况下可以使用，比如在以下情况：

- 删除剩余行程和启动后续程序段之间的时间很短。
- 剩余行程删除的条件有很大的可能性满足。



工作流程

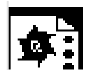
在释放预置剩余行程删除的运动程序段结束处，激活隐含的进刀停止。
因此在程序段结束处，用快速的剩余行程删除中断或者停止轨迹控制运行或定位轴运动。



编程举例

快速剩余行程删除轨迹

```
WHEN $A_IN[1]==1 DO DELDTG
N100 G01 X100 Y100 F1000      ；如果设置输入端，则运动被中断
N110 G01 X…
IF $AA_DELT>50…
```



编程举例

快速轴向剩余行程删除

停止一个编程的定位运动：

```
ID=1 WHEN $A_IN[1]==1 DO MOV[V]=3 FA[V]=700     启动轴
WHEN  $A_IN[2]==1 DO DELDTG(V)       删除剩余行程，用MOV=0停止轴
```

取决于输入端电压，删除剩余行程：

```
WHEN  $A_INA[5]>8000 DO DELDTG(X1)
            ；输入端5电压一旦超过8V，就删除轴X1的剩余行程。
                 轨迹运行继续。
POS[X1]=100 FA[X1]=10 G1 Z100 F1000
```



限制

预置的剩余行程删除

- 在有效的刀具半径补偿时不可以使用。
- 仅在程序段方式有效的同步动作中（没有ID号）
  编程动作。

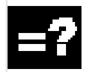


10.4.6 多项式定义，FCTDEF，程序段同步



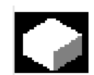
编程

```
FCTDEF(多项式号,LLIMIT,ULIMIT,a0,a1,a2,a3)
```



说明

Polynom-Nr.	3阶多项式序号
LLIMIT	功能值下限
ULIMIT	功能值上限
a_0, a_1, a_2, a_3	多项式系数



功能

使用FCTDEF可以按照 $y=a_0+a_1 \cdot x+a_2 \cdot x^2+a_3 \cdot x^3$
定义3阶多项式。这些多项式由在线刀具补偿FTOC和
求算功能SYNFCT，用于计算主运行变量
（实时变量）的功能值。

这些多项式可以与功能FCTDEF程序段同步，或者通
过系统变量进行定义：

$AC_FCTLL[n]	功能值下限
$AC_FCTUL[n]	功能值上限
$AC_FCT0[n]	a_0
$AC_FCT1[n]	a_1
$AC_FCT2[n]	a_2
$AC_FCT3[n]	a_3
n	多项式序号





说明：

- 系统变量可以由零件程序，或者从一个同步动作写入。
  在写零件程序时，必须通过编程STOPRE考虑程序段同
  步写入。
- 自软件版本SW4起：
  系统变量 $AC_FCTLL[n], $AC_FCTUL[n],
  $AC_FCT0[n] 至 $AC_FCTn[n]
  可以由同步动作改变(非 SINUMERIK FM-NC,
  非 SINUMERIK 840D 带 NCU 571)。

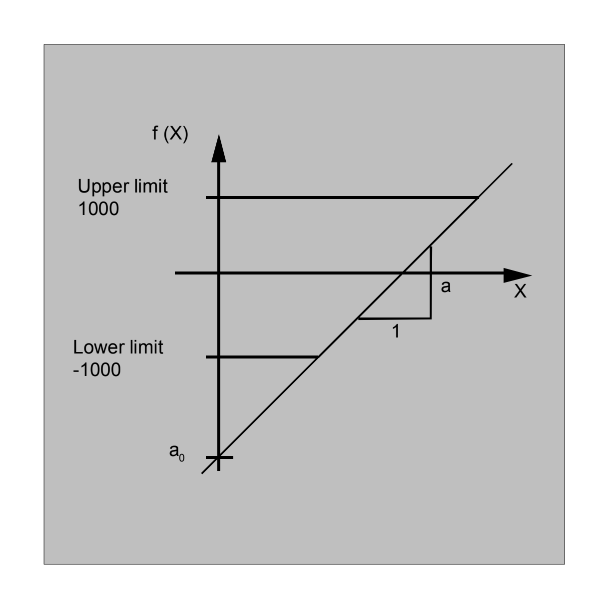
在从同步动作中写入时，多项式系数和功能值界限立即生效。



编程举例

多项式用于直线线段：

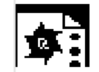
多项式定义如下，其中上限1000，
下限-1000，纵坐标段a_0=$AA_IM[X]，直线升高1：




```
FCTDEF(1, -1000,1000,$AA_IM[X],1)
```



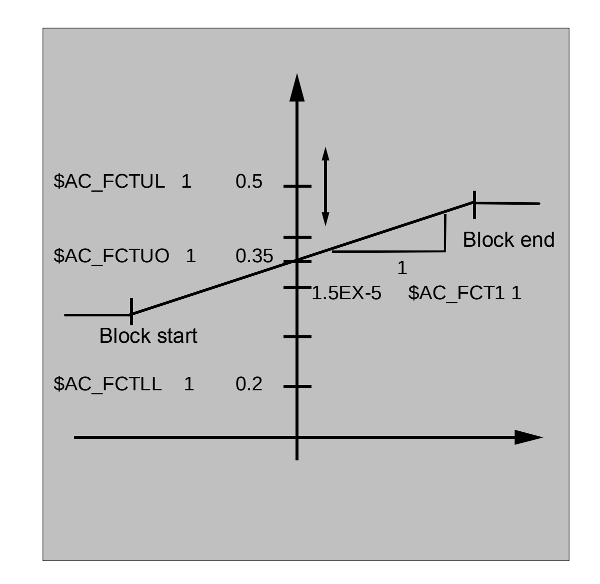
10.4.7 激光器功率控制系统



编程举例

通过变量地多项式定义

激光器功率控制系统是可能的多项式定义应用。
激光器功率控制系统是指：
模拟量输出端的影响，比如取决于轨迹速度。



`$AC_FCTLL[1]=0.2`	多项式系数的定义
`$AC_FCTUL[1]=0.5`	
`$AC_FCT0[1]=0.35`	
`$AC_FCT1[1]=1.5EX-5`	
`STOPRE`	
`ID=1 DO $AC_FCTUL[1]=$A_INA[2]*0.1 +0.35`	在线修改上限
`ID=2 DO SYNFCT(1,$A_OUTA[1],$AC_VACTW)`	
；取决于轨迹速度（存储在$AC_VACTW中），激光器功率控制系统通过模拟量输出端1控制。	



说明

通过SYNFCT使用上面定义的多项式。

10.4.8 求值功能SYNFCT

编程

SYNFCT（多项式号, 实时变量输出端，实时变量输入端）

说明

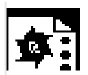
多项式号	用FCTDEF定义的多项式（参见章节“多项式定义”）
实时变量输出端	写实时变量
实时变量输入端	读实时变量



功能

SYNFCT读出处理同步实时变量（比如模拟量输入端，实际值，...），并使用求值多项式（FCTDEF）计算最大3项功能值（比如倍率，速度，轴位置，...）。结果输出到实时变量，并且使用FCTDEF限制上下值(参见章节10.4.7）。

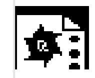
作为实时输出变量，可以选择变量，
- 它们对处理过程有加法影响
- 它们对处理过程有乘法影响
- 它们作为位置补偿进入处理过程
- 它们直接进入处理过程。



应用：

求值功能应用于：
- AC调节（适配控制）
- 激光器功率控制系统
- 位置吸合时

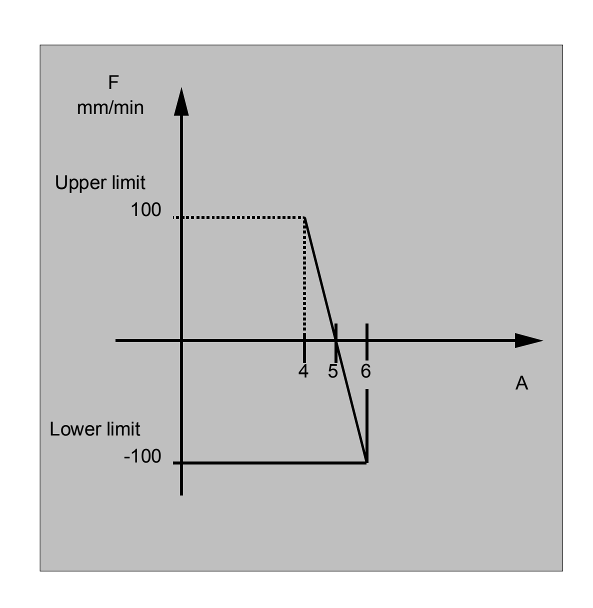
10.4.9 AC调节（加法）



编程举例

编程进给，加法影响

通过X轴（横向进给轴）附加调节一个编程的进给：
进给应在+/- 100 mm/min上下改变，
电流则在5A的工作点波动+/-1A。



1. 多项式定义

确定系数

$y = f(x) = a_0 + a_1x + a_2x^2 + a_3x^3$

a_1 = -100mm/1 min A

a_0 = -(-100)*5 =500

$a_2 = a_3$ = 0 (没有平方项和立方项)

上限 = 100

下限 = -100

由此产生：

```
FCTDEF(1,-100,100,500,-100,0,0)
```

2. 接通AC调节

```
ID=1 DO SYNFCT(1,$AC_VC,$AA_LOAD[x])
```

；通过$AA_LOAD[x] 读出当前的轴负载（%最大的驱动电流），
用上面定义的多项式计算轨迹进给补偿。

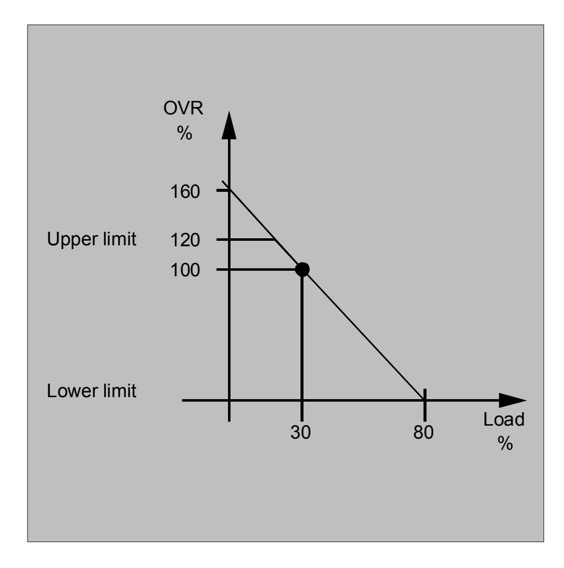
10.4.10 AC调节（乘法）

编程举例

编程的进给，乘法影响

编程的进给应按照乘法影响，此时进给取决于驱动的负载，不应超出一定的界限：

- 当驱动负载为80%时，应停止进给：倍率＝0。
- 当驱动负载为30%时，允许按照编程的进给运行：倍率＝100%。
- 进给速度允许超出最大20%:最大倍率＝120%。



1. 多项式定义

确定系数

$y = f(x) = a_0 + a_1x + a_2x^2 + a_3x^3$

$a_1 = -100\%/(80-30)\% = -2$

$a_0 = 100 + (2*30) = 160$

$a_2 = a_3 = 0$ (没有平方项和立方项)

上限＝120

下限＝0

由此产生：

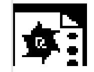
```
FCTDEF(2,0,120,160,-2,0,0)
```

2. 接通AC调节

```
ID=1 DO SYNFCT(2,$AC_OVR,$AA_LOAD[x])
```

；通过$AA_LOAD[x] 读出当前的轴负载（%最大的驱动电流），
用上面定义的多项式计算进给倍率。

10.4.11 距离调节，带有限的补偿



编程举例

距离值的积分计算，带极限范围检查

$AA_OFF_MODE = 1

注意：

超调的调节回路的回路放大与插补节拍的设定相关。

消除方法：读并计算MD，用于插补节拍。

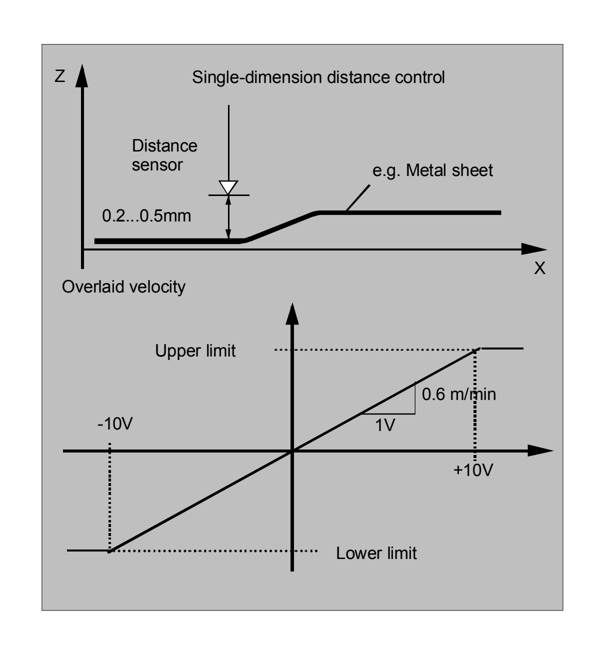
说明：

通过MD32020:JOG_VELO限制超调插补器的速度，

在插补节拍12ms时：

速度：

$$\frac{0.120mm}{0.6ms}/mV \quad 0.6\frac{m}{\min}/V$$



子程序：距离调节 开

%_N_AON_SPF	用于距离调节 开 的子程序
PROC AON	
$AA_OFF_LIMIT[Z]=1	确定极限值
FCTDEF(1, -10, +10, 0, 0.6, 0.12)	多项式定义
ID=1 DO SYNFCT(1,$AA_OFF[Z],$A_INA[3])	距离调节 有效
ID=2 WHENEVER $AA_OFF_LIMIT[Z]<>0 DO $AA_OVR[X] = 0	当超出极限范围时，禁止轴X
RET	
ENDPROC	

子程序：距离调节 关

%_N_AOFF_SPF	
PROC AOFF	用于距离调节的子程序。关
CANCEL(1)	同步动作距离调节删除
CANCEL(2)	极限范围检查删除
RET	
ENDPROC	

主程序：

%_N_MAIN_MPF	
AON	距离调节 开
...	
G1 X100 F1000	
AOFF	距离调节 关
M30	



说明：

在基准坐标系中的位置偏移

使用系统变量$AA_OFF[轴]，通道中的每个轴可以叠加一个运动。它作为基准坐标系中的位置偏移。

如此编程的位置偏移立即叠加到相应的轴，而与该轴是否编程运行无关。

自软件版本SW4起，可以把绝对修调的值（实时变量输出端）限制到设定数据SD 43350:AA_OFF_LIMIT中存储的值。

通过机床数据MD 36750:AA_OFF_MODE确定该距离的叠加方式：

0 比例加权

1 积分加权

使用系统变量$AA_OFF_LIMIT[轴]可以询问（与方向有关）该补偿值是否在此极限范围之内。系统变量可以由同步动作询问，在达到一个极限值时停止轴或者设定报警。

0 补偿值不在极限范围之内

1 在正方向达到补偿值的极限

-1 在负方向达到补偿值的极限



10.4.12 在线刀具补偿FTOC

编程



```
FTOC(多项式号, EV, 长度1_2_3 或者 半径4, 通道, 主轴)
```

说明



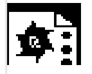
多项式号	使用FCTDEF定义多项式，参见本章中“多项式定义”节。
EV	实时变量，通过说明的多项式计算一个功能值。
长度1_2_3 半径4	长度补偿 ($TC_DP1 至 3)或者半径补偿，计算的功能值加到该值。
通道	通道号，补偿在该通道中生效。在有效通道中的一个补偿在此没有说明。必须在目标通道中接通FTOCON。
主轴	仅在不需要修正有效主轴时予以说明。

功能

FTOC可以提供一个几何轴的叠加运动，按照一个用FCTDEF编程的多项式，与一个基准值相关，比如该基准值可能为一个轴的实际值。
由此您也可以编程模态的在线刀具补偿，或者距离调节作为同步动作。

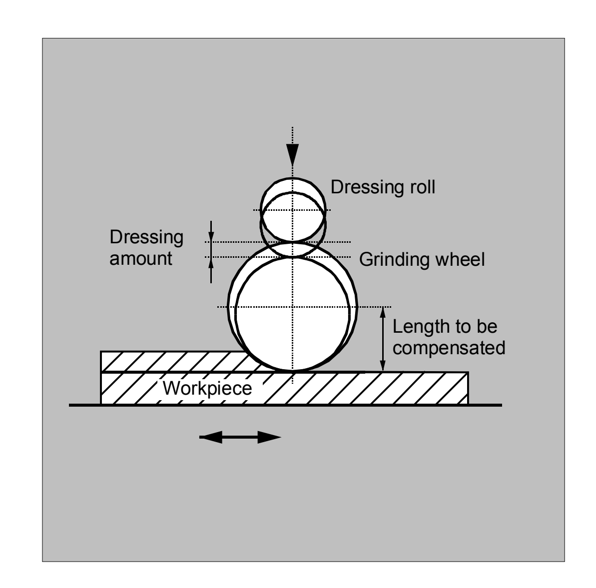
应用

工件的加工，在同一通道中或者不同的通道中调整砂轮（加工通道和调整通道）。
边界条件和确定砂轮的修整适用于FTOC，类似于用PUTFTOCF的在线刀具补偿。
详情参见章节5“刀具补偿”。



编程举例

在该示例中，应该修调当前有效的、正在使用的砂轮的长度。



%_N_ABRICHT_MPF	
FCTDEF(1,-1000,1000,-$AA_IW[V],1)	功能定义。
ID=1 DO **FTOC**(1,$AA_IW[V],3,1)	选择在线刀具补偿： V轴的实际值是多项式1的输入值；结果在通道1中作为补偿值叠加到有效砂轮的长度3。
WAITM(1,1,2)	与加工通道同步
G1 V-0.05 F0.01 G91	用于修整的横向进给运动
G1 V-0.05 F0.02	
...	
CANCEL(1)	取消在线补偿
...	

10.4.13 定位运动



功能

轴可以与零件程序完全异步，由同步动作定位。由同步动作编程定位轴，建议用于循环过程或者由事件控制的过程。由同步动作编程的轴叫做 **指令轴**。

自软件版本SW5起， G70/G71/G700/G710 G代码可以在同步动作中编程。由此可以在同步动作中确定定位任务的尺寸系统。

文献： /PG/ 章节 3“位移尺寸”
/FBSY/ ”启动指令轴”

尺寸系统用G70/G71/G700/G710确定。

通过编程同步动作中的G功能，可以确定同步动作的INCH/METRIC求值系统，而与零件程序文本无关。

举例1	程序环境影响定位轴的定位行程（在同步动作的动作部分没有G功能）。	
N100 R1=0		
N110 G0 X0 Z0		
N120 WAITP(X)		
N130 ID=1 WHENEVER $R==1 DO POS[X]=10		
N140 R1=1		
N150 G71 Z10 F10	Z=10 mm	X=10 mm
N160 G70 Z10 F10	Z=254 mm	X=254 mm
N170 G71 Z10 F10	Z=10 mm	X=10 mm
N180 M30		

举例2

同步动作的动作部分中，G71明确确定定位轴的定位行程（公制），而与程序环境无关。

N100 R1=0		
N110 G0 X0 Z0		
N120 WAITP(X)		
N130 ID=1 WHENEVER $R==1 DO G71 POS[X]=10		
N140 R1=1		
N150 G71 Z10 F10	Z=10 mm	X=10 mm
N160 G70 Z10 F10	Z=254 mm	X=10 mm (X 始终定位到 10 mm)
N170 G71 Z10 F10	Z=10 mm	X=10 mm
N180 M30		

编程举例

禁止编程的轴运行

如果轴运动不用从程序段开始处启动，
则可以把同步动作中的轴的倍率保持到0，
直至所要求的起始时间。

```
WHENEVER    $A_IN[1]==0 DO $AA_OVR[W]=0
            G01 X10 Y25 F750 POS[W]=1500
            FA=1000
```

；一旦数字输入端1＝0，则定位轴被一直停止。

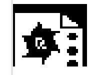
10.4.14 轴定位POS



功能

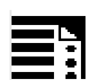
POS[轴]=值

与由零件程序编程相反，定位轴运动对零件程序的执行没有影响。



说明

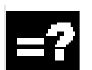
轴：	应当运行的轴名称。
值：	待运行值的说明（根据运行方式）



编程举例

ID=1 EVERY $AA_IM[B]>75 DO **POS[U]=100**
；轴U与运行方式无关，增量运行100 (inch/mm) 或者运行到距系统零点位置 100 (inch/mm)处。
ID=1 EVERY $AA_IM[B]>75 DO **POS[U]=$AA_MW[V]-$AA_IM[W]+13.5**
；轴U运行由实时变量计算的位移。

10.4.15 轴启动/停止MOV

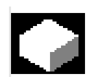


编程

MOV [轴]=值

说明

轴：	应当启动的轴名称。
值：	用于运行/停止运动的启动指令符号确定运动方向。该值的数据类型是INTEGER。
值 >0 (通常方式 +1):	正方向
值 <0 (通常方式 -1):	负方向
值 ==0：	停止轴运动



功能

通过MOV[轴]=值，可以启动一个指令轴，而不对终点位置说明。轴在编程的方向运行，直至通过一个新的运动指令或者定位指令规定另一个运动，或者该轴通过一个停止指令停止。



编程举例

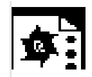
```
... DO MOV[U]=0                      轴U被停止
```



说明

如果用MOV[轴]=0停止一个分度轴，则该轴在下一个分度位置停止。

10.4.16 轴向进给： FA



编程举例

FA[轴]=进给

```
ID=1 EVERY $AA_IM[B]>75 DO POS[U]=100 FA[U]=990
                                   ；固定规定进给值
ID=1 EVERY $AA_IM[B]>75 DO POS[U]=100 FA[U]=$AA_VACTM[W]+100
                                   ；由实时变量构成进给值
```

10.4.17 SW限位开关



功能

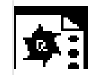
与设定数据$SA_WORKAREA_PLUS_ENABLE相关，考虑指令轴的、G25/G26编程的工作区域限制。
通过零件程序中G功能WALIMON/WALIMOF开关工作区域限制，这对指令轴不起作用。

10.4.18 轴协调



功能

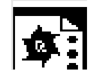
标准情况下，一个轴或者由零件程序运动程序段运动，或者作为定位轴由同步动作运动。
如果同一个轴交替地由零件程序作为轨迹轴或者定位轴运行，或者由同步动作运行，则在两个轴运动之间进行一次协调传送。
如果一个指令轴紧接着由零件程序运行，则它要求一个预处理的重组。它再次决定零件程序加工的中断，与进刀停止类似。



编程举例

X轴可以选择从零件程序运行，
或者从同步动作运行：

N10 G01 **X100** Y200 F1000	在零件程序中编程X轴
…	
N20 ID=1 WHEN $A_IN[1]==1 DO **POS[X]=150** FA[X]=200	当出现数字输入端时，由同步动作启动定位。
…	
CANCEL(1)	撤销选择同步动作
…	
N100 G01 **X240** Y200 F1000	
；X为轨迹轴；由于轴传送，在运动之前出现等待时间，如果是数字输入端1，并且X由同步动作定位。	



编程举例

改变该轴的运行指令：

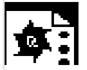
ID=1 EVERY $A_IN[1]>=1 DO POS[V]=100 FA[V]=560
；当数字输入端>=1时，由同步动作启动定位。
ID=2 EVERY $A_IN[2]>=1 DO POS[V]=$AA_IM[V] FA[V]=790
轴随后运行，第二个输入端设置，也就是说在两个同时有效的同步动作中，随着运动执行而跟随终点位置，执行轴V的进给。

10.4.19 设定实际值



功能

在执行 PRESETON (轴，值)时，
不改变当前的轴位置，给它赋值一个新的值。



说明

出自同步动作的PRESETON可以用于：

- 取模回转轴，由零件程序启动
- 所有的指令轴，由同步动作启动

限制：
PRESETON不可用于参加转换的轴。



编程举例

```
WHEN $AA_IM[a] >= 89.5 DO PRESETON(a4,10.5)
                   ；轴a的系统零点在轴正向移动10.5个长度单位（英寸或者毫米）。
```

限制

该轴仅可以错开时间由零件程序或者一个同步动作运动，
因此如果该轴事先在一个同步动作中编程，则在由零件程
序编程一个轴时可能会出现等待时间。
如果交替使用相同的轴，则在两个轴运动之间协调传
送一次。对此零件程序加工必须中断执行。



10.4.20 主轴运动



功能

主轴可以与零件程序完全异步，由同步动作定位。
这种编程方式建议用于循环过程或者由事件控制的过程。



编程举例

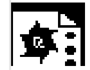
主轴启动/停止/定位

ID=1 EVERY $A_IN[1]==1 DO **M3 S1000**	调节旋转方向和转速
ID=2 EVERY $A_IN[2]==1 DO **SPOS=270**	定位主轴



加工顺序

如果通过同时有效的同步动作给一个主轴规定相互冲突的指令，则从时间的角度来说最后一个主轴指令有效。

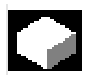


编程举例

调节旋转方向、转速/定位主轴

ID=1 EVERY $A_IN[1]==1 DO M3 S300	调节旋转方向和转速
ID=2 EVERY $A_IN[2]==1 DO M4 S500	给定新的旋转方向和新的转速
ID=3 EVERY $A_IN[3]==1 DO S1000	给定新的转速
ID=4 EVERY ($A_IN[4]==1) AND ($A_IN[1]==0) DO SPOS=0	定位主轴

10.4.21 联动： TRAILON, TRAILOF



功能

DO TRAILON(跟随轴, 引导轴, 耦合系数)	接通联动
DO TRAILOF(跟随轴, 引导轴, 引导轴 2)	关闭联动

在由同步动作接通耦合时，可能是引导轴处于运动当中。在这种情况下，跟随轴加速到给定速度。在速度同步时引导轴的位置是联动的起始位置。联动的功能在章节“轨迹运行特性”中叙述。

激活异步联动：

... DO **TRAILON**(FA, LA, Kf)

带:
- FA: 跟随轴
- LA: 引导轴
- Kf: 耦合系数

取消异步联动：

... DO **TRAILOF**(FA, LA, LA2)

带:
- FA: 跟随轴
- LA: 引导轴
- LA2: 引导轴2，选件



编程举例

$A_IN[1]==0 DO TRAILON(Y,V,1)	如果是数字输入端1，则接通第一个联动组合。
$A_IN[2]==0 DO TRAILON(Z,W,-1)	接通第二个联动组合。
G0 Z10	在相反的轴方向横向进给Z轴和W轴。
G0 Y20	在相同的轴方向横向进给Y轴和V轴。
...	
G1 Y22 V25	叠加联动轴“V”相关和不相关的运动。
...	
TRAILOF(Y,V)	关断第一个联动组合。
TRAILOF(Z,W)	关断第二个联动组合。





必须事先调用TRAILOF功能，从而再次释放一个耦合轴，用于作为通道轴存取。必须保证在通道要求相关轴之前执行TRAILOF。在下面的示例中并非这种情况。

```
...
N50 WHEN TRUE DO TRAILOF(Y, X)
N60 Y100
...
```

在这种情况下，轴没有及时地使能，因为程序段方式有效的同步动作TRAILOF与N60同步有效（参见10.1）。

为了避免相互冲突的情形产生，应按照下面的方式运行：

```
...
N50 WHEN TRUE DO TRAILOF(X, Y)
N55 WAITP(Y)
N60 Y100
```

10.4.22 引导值耦合LEADON, LEADOF



功能

轴向的引导值耦合可以在同步动作中编程，不受限制。

接通轴向引导值耦合：

`...DO **LEADON**(FA,LA,NR)`	带:	FA:	跟随轴
		LA:	引导轴
		NR:	存储的曲线表的序号

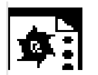
关断轴向引导值耦合：

`...DO **LEADOF**(FA,LA)`	带:	FA:	跟随轴
		LA:	引导轴

为了释放一个待耦合的轴，用于通过同步动作进行存取，必须事先调用功能RELEASE，用于待耦合的跟随轴。

举例:

```
RELEASE (XKAN)
ID=1 every SR1==1 to LEADON(CACH,XKAN,1)
```



编程举例

过程分割

始终通过一个分割工具的工作区运动的带材，应该分割为相同长度的部分。

X轴：　带材运动的轴。WKS

X1-轴：　带材的加工轴, MKS

Y-轴：　分割工具与带材“一起运行”的轴

可以认为分割工具的横向进给与其控制装置由PLC控制。为了确定带材与分割工具的同步性，可以求算PLC接口的信号。

动作　接通耦合，LEADON

　　　关断耦合，LEADOF

　　　设置实际值，PRESETON

```
%_N_SCHERE1_MPF
;$PATH=/_N_WKS_DIR/_N_DEMOFBE_WPD
N100 R3=1500                                   ;一个待分割部件的长度
N200 R2=100000 R13=R2/300
N300 R4=100000
N400 R6=30                                     ；Y轴起始位置
N500 R1=1                                      ；带材轴的起始条件
N600 LEADOF(Y,X)                               ；清除一个可能出现的耦合
N700 CTABDEF(Y,X,1,0)                          ；表定义
N800 X=30 Y=30                                 ；值组对
N900 X=R13 Y=R13
N1000 X=2*R13 Y=30
N1100 CTABEND                                  ；表定义结束
N1200 PRESETON(X1,0)                           ；用于开始的PRESET
N1300 Y=R6 G0                                  ；起始位置Y轴，轴为线性轴
N1400 ID=1 WHENEVER $AA_IW[X]>$R3 DO PESETON(X1,0)
                        ；根据长度R3 PRESET，分割之后新开始
N1500 RELEASE(Y)
N1800 ID=6 EVERY $AA_IM[X]<10  DO LEADON(Y,X,1)
                                               ；在X<10时，Y通过表1耦合到X
N1900 ID=10 EVERY  $AA_IM[X]>$R3-30  DO EADOF(Y,X)
                                               ；>30 在运行的分割长度之前去除耦合
N2000 WAITP(X)
N2100 ID=7 WHEN $R1==1 DO MOV[X]=1             ；带材轴始终处于运动状态
FA[X]=$R4
N2200 M30
```

10.4.23 测量



与零件程序中运动程序段的使用相比较，测量功能可以由同步动作任意开和关。

• 轴向测量，不删除剩余行程：

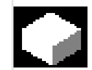
MEAWA[轴]**=(**方式, 触发事件_1, ..._4

• 连续测量，不删除剩余行程：

MEAC[轴]**=(**方式, 测量存储器，触发事件_1, ..._4

有关测量的其它信息：参见章节5，“扩展的测量功能”

10.4.24 设置/删除等待标记：SETM, CLEARM (自软件版本SW 5.2起)



功能

SETM(标记序号)	设置等待标记，用于通道
CLEARM(标记序号)	删除等待标记，用于通道

在同步动作中可以设置或者删除等待标记，比如用于通道间的相互协调。

SETM
指令SETM可以在零件程序中和在同步动作的动作部分写入。该指令设置指令运行通道的标记序号。
CLEARM
指令CLEARM可以在零件程序中和在同步动作的动作部分写入。该指令删除指令运行通道的标记序号。

10.4.25 故障应答

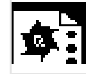


功能

故障应答可以用同步动作编程，通过询问状态变量和释放相应的动作。

对故障的可能应答：

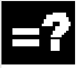
- 轴停止：倍率＝0
- 设置报警：
  使用SETAL可以由同步动作设置循环报警。
- 设置输出端
- 同步动作中所有可能的动作



编程举例

```
ID=67 WHENEVER ($AA_IM[X1]-$AA_IM[X2])<4.567 DO $AA_OVR[X2]=0
            ；如果轴X1和X2之间的安全距离太小，则轴X2停止。
ID=67 WHENEVER ($AA_IM[X1]-$AA_IM[X2])<4.567 DO SETAL(61000)
            ；如果轴X1和X2之间的安全距离太小，则设置报警。
```

10.4.26 运行到固定挡块 FXS 和 FOCON/FOCOF



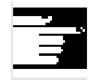
说明	**FXS 和 FOC 在同步动作中**
FXS[轴]	仅在带数字驱动 (VSA, HSA, HLA)的系统中选择
FXST[轴]	改变夹紧扭矩FXST
FXSW[轴]	改变监控窗口FXSW
FOCON[轴]	激活模态有效的扭矩/力－极限
FOCOF[轴]	取消扭矩/力－极限
FOCON/FOCOF	轴的编程在方括号中。允许： – 几何轴 – 名称 – 通道轴 – 名称 – 加工轴 – 名称



功能

用于运行到固定挡块的指令用零件程序指令FXS、FXST和FXSW，在同步动作/工艺循环中编程。
激活可以不带运动进行，扭矩被立即限制。一旦轴由给定值运动，则就对挡块进行监控。

以限制的力矩/力（FOC）运行：
该功能允许通过同步动作，在任何时候修改力矩/力，并可以模态或者程序段相关地进行激活。



说明

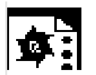
多次选择
一个选择仅允许进行一次。如果由于一次编程出错，在激活之后(FXS[轴]=1)再次调用该功能，则给出报警20092“运行到固定挡块仍有效”。
编程在条件中询问$AA_FXS[]，或者询问一个自身的标志器（这里R1），可以避免该功能的多次激活。
零件程序碎片：

```
N10 R1=0
N20 IDS=1 WHENEVER ($R1==0 AND
$AA_IW[AX3] > 7) DO R1=1 FXST[AX1]=12
```

程序段相关的同步动作：
在接通返回运行期间，通过编程一个程序段相关的同步动作可以运行到固定挡块。
编程举例：

```
N10 G0 G90 X0 Y0
N20 WHEN $AA_IW[X] > 17 DO FXS[X]=1      ；到达X，一个位置大于17毫米
N30 G1 F200 X100 Y110                    ；激活FXS
```

静态的、程序段相关的同步动作：
在静态的、程序段相关的同步动作中可以使用相同的指令FXS, FXST 和FXSW，
如同在正常的零件程序运行中。
所分配的值可以通过一个计算产生。

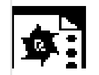


编程举例

碰撞到固定挡块（FXS）

通过一个同步动作释放

Y轴:	；激活静态的同步动作：
N10 IDS=1 WHENEVER (($R1==1) AND ($AA_FXS[y]==0)) DO $R1=0 FXS[Y]=1 FXST[Y]=10 FA[Y]=200 POS[Y]=150	；通过设置$R1=1，给轴Y激活FXS，有效的 ；力矩降低到10%，在挡块方向启动运行
N11 IDS=2 WHENEVER ($AA_FXS[Y]==4) DO FXST[Y]=30	；一旦识别出挡块($AA_FXS[Y]==4)，力矩提 ；高到30%。
N12 IDS=3 WHENEVER ($AA_FXS[Y]==1) DO FXST[Y]=$R0	；在到达挡块之后，力矩控制与R0相关。
N13 IDS=4 WHENEVER (($R3==1) AND ($AA_FXS[Y]==1)) DO FXS[Y]=0 FA[Y]=1000 POS[Y]=0	；与R3相关，撤销选择并返回运行
N20 FXS[Y]=0 G0 G90 X0 Y0	；正常的程序运行：轴Y用于此
N30 RELEASE(Y)	；使能同步动作中运动
N40 G1 F1000 X100	；运动另一个轴
N50	；
N60 GET(Y)	；轴Y再次接收到轨迹组合中



编程举例

激活力矩/力－极限（FOC）

N10 FOCON[X]	；模态方式激活极限
N20 X100 Y200 FXST[X]=15	；X以降低的力矩（15%）运行
N30 FXST[X]=75 X20	；力矩改变到75%，X以此极限力矩运行
N40 FOCOF[X]	；关断力矩极限



10.4.27 在同步动作中确定

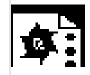


功能

在同步动作中可以读出的系统变量
$AC_TANEB (**T**angent **AN**gel at **E**nd of **B**lock
程序段结束处切线角) 计算一个角，
该角是指当前程序段终点处轨迹切线和编程的后续程
序段起始点处的轨迹切线之间的角度。

切线角始终在范围0.0到180.0度范围内给出正值。
如果在主运行中没有后续程序段，
则给出角度-180.0度。

系统变量$AC_TANEB
不用于读由系统生成的程序段（中间程序段)。系统
变量$AC_BLOCKTYPE用于区别是否与编程的程序
段（主程序段）有关。



编程举例

```
ID=2 EVERY $AC_BLOCKTYPE==0 DO $R1 = $AC_TANEB;
```

10.4.28 确定当前的倍率



功能

当前的倍率
（NC部分）可以用系统变量：

$AA_OVR 轴向倍率
$AC_OVR 轨迹倍率
在同步动作中读写。
由PLC给定的倍率用于读出：

$AA_PLC_OVR 轴向倍率
$AC_PLC_OVR 轨迹倍率。

最后生成的倍率

用于读出系统变量中同步动作：

$AA_TOTAL_OVR　　轴向倍率

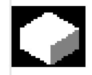
$AC_TOTAL_OVR　　轨迹倍率。

计算最后生成的倍率，作为：

$AA_OVR * $AA_PLC_OVR　　或者

$AC_OVR * $AC_PLC_OVR

10.4.29 通过同步动作的时间占用计算负荷



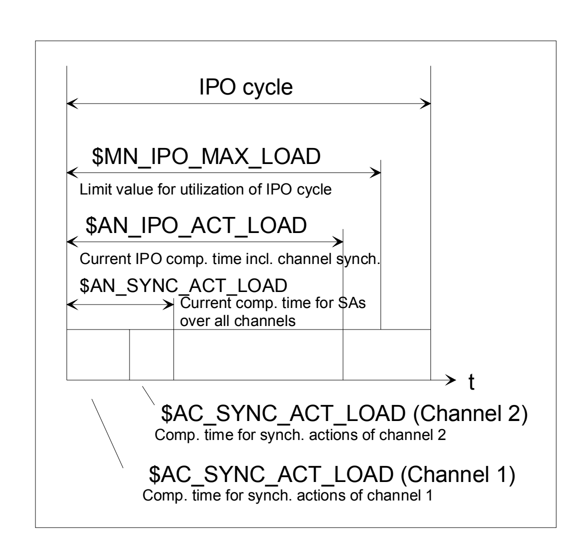
功能

在一个插补节拍中不仅要译出同步动作，而且也必须要由NC计算出运动。利用后面介绍的系统变量，同步动作可以通过插补节拍中同步动作的实际时间分量和位置调节器的计算时间进行了解。

只有在机床数据$MN_IPO_MAX_LOAD大于0时，变量才有有效值。在其它情况下，变量始终说明总的计算时间。总的计算时间由以下构成：

- 同步动作时间，
- 位置调节时间，以及
- 其余IPO计算时间

这些变量始终包含先前的IPO节拍的值。



$AN_IPO_ACT_LOAD	当前的IPO计算时间（包括所有通道中的同步动作）。
$AN_IPO_MAX_LOAD	最长的IPO计算时间（包括所有通道中的同步动作）。
$AN_IPO_MIN_LOAD	最短的IPO计算时间（包括所有通道中的同步动作）。
$AN_IPO_LOAD_PERCENT	当前的IPO计算时间，与IPO节拍比较（%）。
$AN_SYNC_ACT_LOAD	当前的计算时间，用于所有通道的同步动作
$AN_SYNC_MAX_LOAD	最长的计算时间，用于所有通道的同步动作

$AN_SYNC_TO_IPO	在总的IPO计算时间中（通过所有通道）总的同步动作的百分比。
$AC_SYNC_ACT_LOAD	通道中同步动作的当前计算时间
$AC_SYNC_MAX_LOAD	通道中同步动作的最长的计算时间
$AC_SYNC_AVERAGE_LOAD	通道中同步动作的平均的计算时间
$AN_SERVO_ACT_LOAD	位置调节器的当前的计算时间
$AN_SERVO_MAX_LOAD	位置调节器的最长的计算时间
$AN_SERVO_MIN_LOAD	位置调节器的最短的计算时间

过载通知：

通过MD
$MN_IPO_MAX_LOAD可以设定，自多少IPO－总－计算时间（IPO节拍的%）起应设置系统变量$AN_IPO_LOAD_LIMIT为TRUE（真)。如果当前的负载又再次低于极限，则该变量再次置为FALSE（假)。如果MD为0，则整个诊断功能无效。
通过求值$AN_IPO_LOAD_LIMIT，
用户可以自己确定一个方案，从而避免平面超程。
系统变量：

$AN_SERVO_MAX_LOAD	位置调节器的最长的计算时间
$AN_SERVO_MIN_LOAD	位置调节器的最短的计算时间
$AN_IPO_MAX_LOAD	最长的IPO计算时间（包括所有通道中的同步动作）。
$AN_IPO_MIN_LOAD	最短的IPO计算时间（包括所有通道中的同步动作）。
$AN_SYNC_MAX_LOAD	最长的计算时间，用于所有通道的同步动作
$AC_SYNC_MAX_LOAD	通道中同步动作的最长的计算时间

由同步动作可以描述。这些变量在每次写存取时复位到当前的负载，而与所写入的值无关。

编程举例

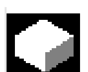
$MN_IPO_MAX_LOAD = 80

MD:IPO节拍的使用极限
只要 $AN_IPO_LOAD_PERCENT > 80 %, 则 $AN_IPO_LOAD_LIMIT 就置为 TRUE （真）。

```
N01   $AN_SERVO_MAX_LOAD=0
N02   $AN_SERVO_MIN_LOAD=0
N03   $AN_IPO_MAX_LOAD=0
N04   $AN_IPO_MIN_LOAD=0
N05   $AN_SYNC_MAX_LOAD=0
N06   $AC_SYNC_MAX_LOAD=0
N10   IDS=1 WHENEVER $AN_IPO_LOAD_LIMIT == TRUE DO M4711 SETAL(63111)
N20   IDS=2 WHENEVER $AN_SYNC_TO_IPO > 30 DO SETAL(63222)
N30   G0 X0 Y0 Z0
...
N999  M30
```

如果超出总的使用极限，则第一个同步动作产生一个辅助功能输出，并且有一个报警。
如果同步动作在IPO计算时间中的分量（通过所有通道）超出30%，则第二个同步动作产生一个报警。

10.5 工艺循环



功能

作为同步动作中的动作，也可以调用仅由功能组成的程序，这些功能也可以允许作为同步动作中的动作。由此构成的程序称作工艺循环。

工艺循环可以作为子程序存储在控制系统中。对于用户，调用如同子程序。不可以进行参数传送。

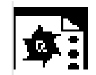
在一个通道中可以并行处理几个工艺循环或者动作。

程序结束以M02/M17/M30/RET 编程。
每个程序段最多可以编程一个轴运行。



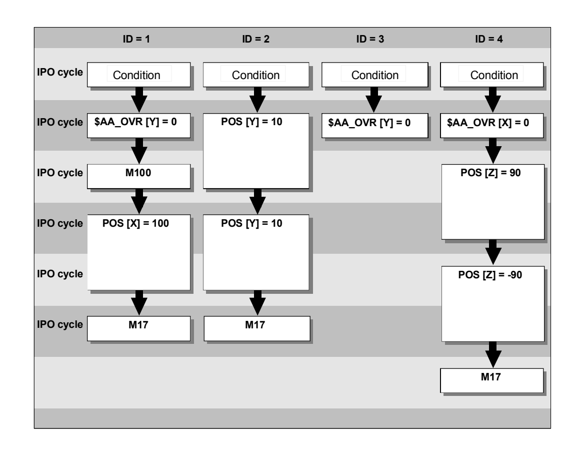
应用

工艺循环作为轴程序：每个工艺循环仅控制一个轴。
因此不同的轴运行可以在同一个插补节拍中由事件控
制启动。该零件程序在特殊情况下
仅用于管理同步动作。



编程举例

通过设置数字输入端，启动轴程序。



主程序：

`ID=1 EVERY $A_IN[1]==1 DO ACHSE_X`	如果输入端1为1，则启动轴程序X
`ID=2 EVERY $A_IN[2]==1 DO ACHSE_Y`	如果输入端2为1，则启动轴程序Y
`ID=3 EVERY $A_IN[3]==1 DO $AA_OVR[Y]=0`	如果输入端3为1，则轴Y的倍率置为0。
`ID=4 EVERY $A_IN[4]==1 DO ACHSE_Z`	如果输入端4为1，则启动轴程序Z
`M30`	

工艺循环 ACHSE_X:

```
$AA_OVR[Y]=0
M100
POS[X]=100 FA[X]=300
M17
```

工艺循环 ACHSE_Y:

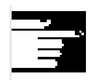
```
POS[Y]=10 FA[Y]=200
POS[Y]=-10
M17
```

工艺循环 ACHSE_Z:

```
$AA_OVR[X]=0
POS[Z]=90 FA[Z]=250
POS[Z]=-90
M17
```



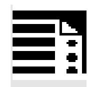
一旦条件满足，则启动工艺循环。在定位轴中，需要几个IPO节拍用于执行。其它的功能（OVR）执行一个节拍。
在工艺循环中，程序段按顺序执行。



说明

如果在同一个插补节拍中调用几个动作，这几个动作相互之间排斥，则启动同步动作中用更高ID号调用的动作。

10.5.1 禁止，释放，中断： LOCK, UNLOCK, RESET



编程

LOCK (n, n, ...)	禁止工艺循环，其有效动作被中断
UNLOCK (n, n, ...)	释放工艺循环
RESET (n, n, ...)	中断工艺循环，其有效动作被中断
n	同步动作的识别号



功能

工艺循环的过程可以由同步动作或者由一个工艺循环禁止、释放和中断。

禁止工艺循环，LOCK

工艺循环用LOCK由一个其它的同步动作或者由一个工艺循环禁止。

举例:

```
N100 ID=1 WHENEVER $A_IN[1]==1 DO M130
...
N200 ID=2 WHENEVER $A_IN[2]==1 DO LOCK(1)
```

释放工艺循环，UNLOCK

禁止的工艺循环可以用UNLOCK由一个其它的同步动作/工艺循环再次释放。用UNLOCK可以在当前的位置继续，同样在一个中断的定位过程中。

举例:

```
N100 ID=1 WHENEVER $A_IN[1]==1 DO M130
...
N200 ID=2 WHENEVER $A_IN[2]==1 DO LOCK(1)
...
N250 ID=3 WHENEVER $A_IN[3]==1 DO UNLOCK(1)
```

中断工艺循环，RESET

工艺循环可以用RESET由一个其它的同步动作或者由一个工艺循环中断。

举例:

```
N100 ID=1 WHENEVER $A_IN[1]==1 DO M130
...
N200 ID=2 WHENEVER $A_IN[2]==1 DO RESET(1)
```

PLC一侧闭锁

模态有效的同步动作，带ID号**n=1…64**，可以由PLC闭锁。因此相关的条件不再计算，相应的功能在NCK中禁止执行。

使用一个PLC的接口信号可以禁止所有同步动作。



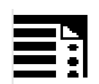
说明

一个编程的同步动作按照标准激活，可以通过机床数据防止改写/禁止。

应用：

由机床制造商确定的同步动作应该不可以由最终用户进行修改。

10.6 删除同步动作：CANCEL



编程

`CANCEL(n,n,...)`	删除同步动作
`n`	同步动作的识别号

说明

带名称ID(S)=n的模态有效的同步动作仅可以用
CANCEL直接由零件程序删除。

举例:

`N100 ID=2 WHENEVER $A_IN[1]==1 DO M130`	
`...`	
`N200` **`CANCEL(2)`**	删除同步动作No.2



说明

由一个删除的同步动作启动的、仍然在运行的动作根
据编程结束。

10.7 边界条件：

- **Power On**（上电）
  执行上电，没有同步动作有效。
  但是静态的同步动作在上电时可以立即用一个由
  PLC启动的异步子程序（ASUP）激活。

- 工作方式变换
  使用关键字IDS激活的同步动作仍然有效，与工
  作方式变换 无关。
  所有其它的同步动作在工作方式变换时无效（比
  如轴定位），并且用再定位和换回到自动方式时
  再次生效。

- **Reset**
  用NC复位所有通过同步动作启动的动作停止。静态的同步动作仍然有效。由它们可以启动新的动作。
  使用同步动作中的RESET，或者一个工艺循环中的RESET，可以复位一个模态有效的同步动作。
  如果一个同步动作复位，但是同时同步动作激活的定位轴运动却仍有效，则停止定位轴运行。
  所执行的WHEN类型的同步动作在RESET之后不再执行。

RESET之后的性能		
同步动作/工艺循环	**模态有效/程序段方式**	**静态 (IDS)**
	有效的动作停止，同步动作删除	有效动作停止，工艺循环复位
轴/定位的主轴	运动被停止	运动被停止
转速控制的主轴	$MA_SPIND_ACTIVE_AFTER_RESET==1: 主轴仍然有效 $MA_SPIND_ACTIVE_AFTER_RESET==0: 主轴停止	$MA_SPIND_ACTIVE_AFTER_RESET==1:主轴仍然有效 $MA_SPIND_ACTIVE_AFTER_RESET==0: 主轴停止
引导值耦合	$MC_RESET_MODE_MASK, Bit13 == 1: 引导值耦合仍然有效 $MC_RESET_MODE_MASK, Bit13 == 0: 引导值耦合废除	$MC_RESET_MODE_MASK, Bit13 == 1:引导值耦合仍然有效 $MC_RESET_MODE_MASK, Bit13 == 0:引导值耦合废除
测量过程	由同步动作启动的测量过程停止	由静态同步动作启动的测量过程停止

- **NC-Stop**
  静态 同步动作在NC停止时
  仍然有效。由静态同步动作启动的运动没有被停止。
  属于有效程序段的局部程序的同步动作仍然有效，由此启动的运动被停止。

- 程序结束
  程序结束 和同步动作相互没有影响。
  运行的同步动作在程序结束之后也被结束。
  在M30程序段中有效的同步动作在M30程序段中仍然有效。如果要求这样，则同步动作必须在程序结束之前用CANCEL（参见前面的章节）停止。

程序结束之后的性能		
同步动作/工艺循环	模态方式和程序段方式被停止	静态(IDS)保持不变
轴/定位的主轴	M30延迟，直至轴/主轴停止	运动继续进行
转速控制的主轴	程序结束 $MA_SPIND_ACTIVE_AFTER_RESET==1: 主轴仍然有效 $MA_SPIND_ACTIVE_AFTER_RESET==0: 主轴停止 在工作方式变换时主轴保持有效	主轴保持有效
引导值耦合	$MC_RESET_MODE_MASK, Bit13 == 1: 引导值耦合仍然有效 $MC_RESET_MODE_MASK, Bit13 == 0: 引导值耦合废除	由静态同步动作启动的耦合保持不变。
测量过程	由同步动作启动的测量过程停止	由静态同步动作启动的测量过程保持有效。

- 程序段搜索
  在程序段搜索期间
  所找到的同步动作被汇聚，并在NC启动时计算，有些情况下启动相关的动作。
  在程序段搜索期间，静态同步动作也有效。
  如果在程序段搜索时用FCTDEF编程的多项式系数被找到，则它直接变为有效。

- 通过异步子程序中断程序
  ASUP-开始：
  模态有效的和静态的运行同步动作保持有效，在异步子程序时也有效。
  ASUP-结束：
  如果异步子程序没有用Repos(再定位)继续，则在异步子程序中改变的模态和静态运行同步动作在主程序中继续有效。

- 再定位

  在再定位之后 REPOS在中断程序段中有效的同步动作再次生效。

  由异步子程序中修改的模态同步动作在REPOS (再定位）之后，在执行剩余程序段时不再生效。

  由FCTDEF编程的多项式系数不会受到异步子程序和REPOS的影响。与在哪里编程无关，它们在异步子程序中和在主程序中在执行REPOS之后任何时间都可以使用。

- 用**CANCEL**撤销选择

  如果用**CANCEL**撤销选择一个有效的同步动作，则有效的动作不受影响。定位运行结束，如同编程一样。

  用指令CANCEL可以停止一个模态或者静态有效的同步动作。

  如果一个同步动作停止，但是同时由此激活的定位轴运动却仍有效，则结束定位轴运行。如果不希望如此，则可以在CANCEL指令之前制动带轴向剩余行程删除的轴运动：

举例:

```
ID=17 EVERY $A_IN[3]==1 DO  POS[X]=15 FA[X]=1500      ；启动定位轴运行
...
WHEN ... DO DELDTG(X)                                 ；结束定位轴运行
CANCEL(1)
```

■

摆动



11.1 异步摆动

命令解释

OSP1 [Achse]=	换向点1的位置
OSP2 [Achse]=	换向点2的位置
OST1 [Achse]= OST2 [Achse]=	在换向点处的保持时间，单位秒
FA[Achse]=	摆动轴进给
OSCTRL [Achse]=	（设置选件，复位选件）
OSNSC [Achse]=	修光冲程数
OSE [Achse]=	终点位置
OS [Achse]=	1＝接通摆动轴；0＝关断摆动轴



功能

一个摆动轴在两个换向点1和2之间以给定的进给来回摆动，直至摆动运动关断。

在摆动运行期间其它的轴可以任意插补。

通过一个轨迹运动或者用一个定位轴，可以达到一个连续的横向进给。但是在此，在摆动运动和横向进给运动之间没有关联。

摆动轴

对于摆动轴 适用于：

- 每个轴可以作为摆动轴使用。
- 可以同时有几个摆动轴有效
  （最大：定位轴个数）。
- 对于摆动轴，与程序中实际有效的G指令无关，
  线性插补 G1 一直有效。

摆动轴可以：

- 是动态转换的输入轴，
- 是龙门轴和联动轴时的引导轴，
- 运行
  – 没有冲击限制 (BRISK) 或者
  – 带冲击限制 (SOFT) 或者
  – 带折弯的加速曲线
  （如同定位轴）。

摆动换向点

在确定摆动位置时必须考虑当前的偏移：

- 绝对尺寸

OSP1[Z]=Wert

换向点位置＝偏移之和+编程的值

- 相对尺寸

OSP1[Z]=IC(值)

换向点位置＝换向点1+编程值

举例:

```
N10 OSP1[Z]=100 OSP2[Z]=110
.
.
N40 OSP1[Z]=IC(3)
```

异步摆动特性

- 异步摆动 为轴专用，并且超出程序段界限生效。
- 通过零件程序，保证一个程序段同步的摆动运动
  开启。
- 几个轴的共同插补和摆动距离的叠加是不可能的。

设定数据

异步摆动所要求的设定数据可以在零件程序中设定。

如果在零件程序中直接描述设定数据，则修改在进刀
时就已经生效。通过进刀停止 STOPRE
可以达到同步性能。

举例:

摆动，带在线修改换向点位置

```
$SA_OSCILL_REVERSE_POS1[Z]=-10
$SA_OSCILL_REVERSE_POS2[Z]=10

G0 X0 Z0
WAITP(Z)

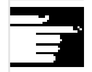
ID=1 WHENEVER $AA_IM[Z] < $$AA_OSCILL_REVERSE_POS1[Z] DO $AA_OVR[X]=0
ID=2 WHENEVER $AA_IM[Z] < $$AA_OSCILL_REVERSE_POS2[Z] DO $AA_OVR[X]=0
                                        ；如果摆动轴的实际值
                                        ；已经超过换向点
                                        ；横向进给轴保持。
OS[Z]=1 FA[X]=1000 POS[X]=40            ；接通摆动
OS[Z]=0                                 ；关断摆动
M30
```



对单个功能的说明

通过后面的地址，可以接通执行相应的NC程序，并且由零件程序影响异步摆动。
编程的值在主运行中与程序段同步登记到相应的设定数据，并且直至下一次修改一直保持有效。

开启，关断摆动: OS
OS[轴] = 1:开启
OS[轴] = 0:关断



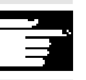
WAITP(轴):

- 如果应该用一个几何轴摆动，则它必须用WAITP使能用于摆动。
- 在摆动结束之后，用该指令把摆动轴再次作为定位轴登记，并且可以再次正常使用。

在换向点的停留时间：
OST1, **OST2**

停留时间	在精确停止范围内的运动特性，在换向点处
-2	继续插补，无需等待精确停止
-1	粗等待精确停止
0	精等待精确停止
>0	精等待精确停止，并且接着等候停留时间

停留时间的单位与通过 G4 编程的停留时间一致。



说明

带运行同步动作和停留时间"OST1/OST2”摆动
在设定的停留时间结束之后，在摆动时开始内部的程序段转换（在轴新的剩余行程时可见）。在程序段转换时检查关断功能。根据设定的控制方式（用于运动过程"OSCTRL”），确定关断功能。

该时间特性可以通过进给倍率影响。
在有些情况下，可以在启动修光冲程之前或者返回终点之前，执行一次摆动冲程。
由此会产生一个印象，即它改变了其关断特性。但是这并非该情况。

设定进给FA
作为进给速度，适用于定义的定位轴的进给速度。
如果没有定义进给速度，则机床数据中存储的值适用。

定义运动过程： OSCTRL
该运动过程的控制装置位置用设定选件和复位选件设置。

复位选件
该选件被关断（仅在事先作为设定选件开通时才可以）。

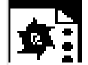
设定选件
转换该选件。在编程 OSE (最终位置)时隐含选件4生效。

选件值	意义
0	在下一个换向点处关断摆动运动时停止（预设定）；仅可以复位值1和2
1	在换向点1处关断摆动运动时停止
2	在换向点2处关断摆动运动时停止
3	如果没有编程修光冲程，则在关断摆动运动时不返回换向点运行。
4	在修光后返回终点位置
8	如果通过剩余行程删除停止摆动运行：则紧接着执行修光冲程，有时要返回终点运行。
16	如果通过剩余行程删除停止摆动运行： 则如同在关断时一样返回相应的换向点。
32	修改的冲程仅从下一个换向点开始才有效
64	FA = 0:位移叠加有效 FA 0:速度叠加有效
128	在回转轴DC时（最短的行程）
256	0＝修光冲程作为双冲程执行。（标准）1＝修光冲程作为单个冲程执行。

多个选件通过正号一个个连接在一起。

举例:

```
OSCTRL[Z] = (1+4,16+32+64)
```



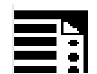
编程举例

摆动轴Z应该在10和100之间摆动。换向点1以精准停返回，换向点2以粗准停返回。应该用进给250进行摆动轴的加工。在加工结束处应该执行3个修光冲程并且用摆动轴控制终点位置200。
横向进给轴的进给是1，在X方向的横向进给在15处结束。

WAITP(X,Y,Z)	输出端位置
G0 X100 Y100 Z100	转换定位轴运行
N40 WAITP(X,Z)	
N50 OSP1[Z]=10 OSP2[Z]=100 -> -> OSE[Z]=200 -> -> OST1[Z]=0 OST2[Z]=-1 -> -> FA[Z]=250 FA[X]=1 -> -> OSCTRL[Z]=(4,0) -> -> OSNSC[Z]=3 -> N60 OS[Z]=1	换向点1，换向点2 终点位置 在U1处的停留时间：精准停； 在U2处的停留时间：粗准停 摆动轴进给，横向进给轴 设定选件 3个修光冲程 启动摆动
N70 WHEN $A_IN[3]==TRUE -> -> DO DELDTG(X)	剩余行程删除
N80 POS[X]=15	X轴输出位置
N90 POS[X]=50	
N100 OS[Z]=0	停止摆动
M30	

->可以在一个程序段中编程。

11.2 通过同步动作控制的摆动



编程

1. 确定摆动参数
2. 定义运动同步动作
3. 分配轴，确定横向进给

摆动参数

OSP1[Pendelachse]=	换向点1的位置
OSP2[Pendelachse]=	换向点2的位置
OST1[Pendelachse]=	在换向点1处的保持时间，单位秒
OST2[Pendelachse]=	在换向点2处的保持时间，单位秒
FA[Pendelachse]=	摆动轴进给
OSCTRL[Pendelachse]=	设定选件或者
OSNSC[Pendelachse]=	修光冲程数
OSE[Pendelachse]=	终点位置
WAITP(Pendelachse)	使能用于摆动的轴

轴分配，横向进给

```
OSCILL[摆动轴] = (横向进给轴1, 横向进给轴2, 横向进给轴3)
POSP[横向进给轴] = (终点位置, 分度长度, 方式)
```

OSCILL	给摆动轴分配横向进给轴
POSP	确定总横向进给和分度横向进给（参见章节3）
Endpos	横向进给轴最终位置，在所有分度横向进给开始运行之后。
Teillänge	在换向点/换向区处的分度横向进给大小
Modus	总的横向进给划分为几个分度横向进给 0＝两个相同大小的剩余步（预设定）； 1＝所有分度横向进给大小相同

运动同步动作

WHEN… … DO	当...时，然后...
WHENEVER … DO	总是当...时，然后...

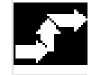


通过同步动作控制摆动

在这种摆动方式，仅在换向点或者在定义的换向范围之内允许一个横向进给运动。

根据具体的要求，可以在横向进给期间

- 继续执行摆动运动，或者
- 停止摆动运动，直至横向进给完全执行。



工作流程

1. 确定摆动参数

在运动程序段之前（该程序段包含横向进给轴和摆动轴的分配，以及确定横向进给），必须确定摆动的参数（参见“异步摆动”）。

2. 确定运动同步动作

通过同步条件进行：

- **抑制横向进给**, 直至摆动轴位于一个换向区之内 (ii1, ii2)，或者在一个换向点 (U1, U2) 处。
- 在横向进给期间在换向点处停止摆动运动。
- 在结束分度横向进给之后启动摆动运动。
- 确定下一个分度横向进给的启动 。

3. 分配摆动轴和横向进给轴，以及确定总的横向进给和分度横向进给。

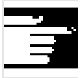


分配摆动轴和横向进给轴: OSCILL

```
OSCILL[摆动轴] = (横向进给轴1, 横向进给轴2, 横向进给轴3)
```

使用指令OSCILL进行轴分配和启动摆动运动。

一个摆动轴最多分配3个横向进给轴。



在启动摆动之前必须已经确定轴性能的同步条件。



确定横向进给: POSP

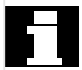
```
POSP[横向进给轴] = (终点位置, 分度部件, 方式)
```

通过指令POSP通知控制系统：

- 总的横向进给（通过终点位置）
- 在换向点或者在换向区处其分度横向进给的大小。
- 在到达终点位置时（通过方式）分度横向进给特性

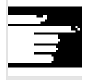
方式＝0	对于两个最后的分度横向进给，划分剩余的位移，直至目标点在2个相同大小的剩余步（预设定）。
方式＝1	所有分度横向进给大小相同。它们由总的横向进给计算。





同步动作

后面所执行的运动同步动作完全用于摆动。
您可以找到方案举例，用于满足您作为编制用户专用的摆动运动的模块而提出的各个要求。

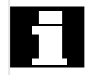


在单个情况下，同步条件也可以另外编程。



关键字

WHEN … DO …	当…时，然后…
WHENEVER … DO	总是当…时，然后…

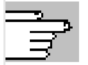


使用后面详细描述的语言方法，您可以实现以下的功能：

1. 在换向点处的横向进给。
2. 在换向区的横向进给。
3. 在两个换向点处的横向进给。
4. 在换向点处停止摆动运动。
5. 再次启动摆动运动。
6. 分度横向进给不要启动太早。

所有这里举例说明的同步动作适用于以下设定：

- 换向点1 < 换向点2
- Z = 摆动轴
- X = 横向进给轴



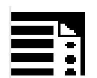
更详细的运动同步动作说明参见章节11.3。



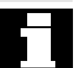
换向区中横向进给

横向进给运动
应该在一个换向区之内开始，在到达换向点之前。

该同步动作阻止横向进给运动，直至摆动轴位于一个
换向区之内。



在所进行的假设条件下（见上面）产生以下的指令：



换向区1：

```
WHENEVER $AA_IM[Z]>$SA_OSCILL_REVERSE_POS1[Z]+ii1 DO $AA_OVR[X]=0
```

如果	当前的、在MKS中的摆动轴的位置
大于	换向区1的开始，
则	设定横向进给轴的轴向倍率到0%。

换向区2：

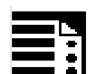
```
WHENEVER $AA_IM[Z] <$SA_OSCILL_REVERSE_POS2[Z]+ii2 DO $AA_OVR[X]=0
```

如果	当前的、在MKS中的摆动轴的位置
小于	换向区2的开始，
则	设定横向进给轴的轴向倍率到0%。



在换向点处的横向进给

只要摆动轴没有到达换向点，就不进行横向进给轴的运动。



在所给定的假设条件下（见上面）产生以下的指令：

换向点1：

```
WHENEVER $AA_IM[Z]<>$SA_OSCILL_REVERSE_POS1[Z] DO $AA_OVR[X]=0 ->
-> $AA_OVR[Z]=100
```

如果	当前的、在MKS中的摆动轴Z的位置
大于或者小于	换向点1的位置，
则	设定横向进给轴的轴向倍率到0%。
并且	摆动轴Z的轴向倍率到100%。

换向点2：

用于换向点2：

```
WHENEVER $AA_IM[Z]<>$SA_OSCILL_REVERSE_POS2[Z] DO $AA_OVR[X]=0 ->
-> $AA_OVR[Z]=100
```

如果	当前的、在MKS中的摆动轴Z的位置
大于或者小于	换向点2的位置，
则	设定横向进给轴的轴向倍率到0%。
并且	摆动轴Z的轴向倍率到100%。



在换向点处停止摆动运动

在换向点处摆动轴停止，同时开始横向进给运动.
如果横向进给运动完全执行，则继续执行摆动运动。

如果横向进给运动通过一个事先进行的、仍然有效的同步动作停止，则可以同时使用该同步动作，启动横向进给运动。



在所给定的假设条件下（见上面）产生以下的指令：

换向点1：

```
WHENEVER $SA_IM[Z]==$SA_OSCILL_REVERSE_POS1[Z]DO $AA_OVR[Z]=0 ->
-> $AA_OVR[X] = 100
```

如果	当前的、在MKS中的摆动轴的位置
相同于	换向位置1，
则	设定摆动轴的轴向倍率到0%。
并且	横向进给轴的轴向倍率到100%。

换向点2：

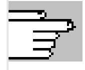
```
WHENEVER $SA_IM[Z] ==$SA_OSCILL_REVERSE_POS2[Z]DO $AA_OVR[Z]= 0 ->
-> $AA_OVR[X]=100
```

如果	当前的、在MKS中的摆动轴的位置
相同于	换向位置2，
则	设定摆动轴的轴向倍率到0%。
并且	横向进给轴的轴向倍率到100%。

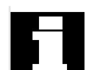


换向点的在线计算

如果在比较符的右侧有一个用**$$**标记的主运行变量，
则计算IPO节拍中 中的两个变量，并相互进行比较。

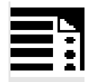


详细的信息参见章节“运动同步动作”。



再次启动摆动运动

如果分度横向进给运动已经结束，则使用同步动作，
继续摆动轴的运动。



在所给定的假设条件下（见上面）产生以下的指令：

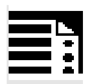
```
WHENEVER $AA_DTEPW[X]==0 DO $AA_OVR[Z]= 100
```

如果	在WKS中横向进给轴X的剩余行程
相同于	为零，
则	设定摆动轴的轴向倍率 到100%。



下一个分度横向进给

在进行横向进给之后，必须防止过早地启动下一个分度横向进给。
对此使用一个通道专用地标记器 ($AC_MARKER[Index])，它位于分度横向进给的末端(分度剩余行程 ≡ 0)，
并且在离开换向区时删除。然后用一个同步动作阻止下一个横向进给运动。



在所给定的假设条件下（见上面）产生以下的指令，比如用于换向点1：

1. 设定标记器：

```
WHENEVER $AA_DTEPW[X] == 0 DO $AC_MARKER[1]=1
```

如果	在WKS中横向进给轴X的分度横向进给剩余行程
相同于	为零，
则	设置标记器（带变址1）到1。

2. 删除标记器

```
WHENEVER $AA_IM[Z]<>$SA_OSCILL_REVERSE_POS1[Z] DO $AC_MARKER[1]=0
```

如果	当前的、在MKS中的摆动轴Z的位置
大于或者小于	换向点1的位置，
则	设定标记器1到0。

3. 阻止横向进给

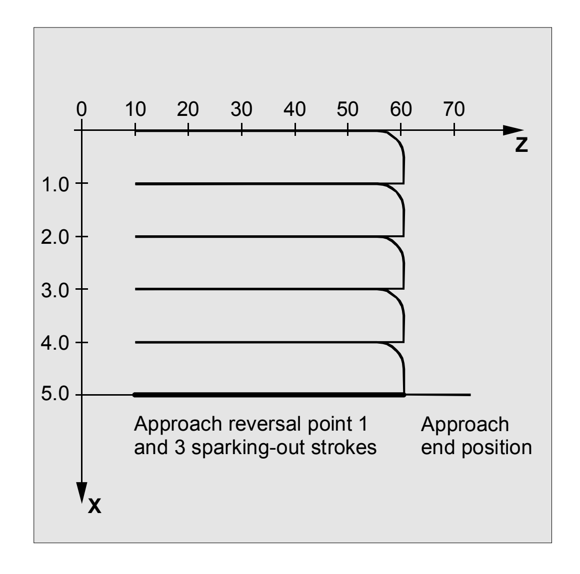
```
WHENEVER $AC_MARKER[1]==1 DO $AA_OVR[X]=0
```

如果	标记器1
相同于	1，
则	设定横向进给轴的轴向倍率到0%。



编程举例

在换向点1不应进行横向进给。在换向点2横向进给应该已经在换向点2之前距离ii2处进行，并且在换向点处摆动轴不等待分度横向进给结束。轴Z是摆动轴，轴X为横向进给轴.



程序分段

1. 摆动参数

`DEF INT ii2`	定义变量，用于换向区2
`OSP1[Z]=10 OSP2[Z]=60`	定义换向点1和2
`OST1[Z]=0 OST2[Z]=0`	换向点1：精准停换向点2：精准停；
`FA[Z]=150 FA[X]=0.5`	摆动轴Z进给，横向进给轴X进给
`OSCTRL[Z]=(2+8+16,1)`	在换向点2处关断摆动运动；在RWL之后修光并返回终点位置；在RWL之后返回相应的换向位置
`OSNC[Z]=3`	3个修光冲程
`OSE[Z]=70`	终点位置＝70
`ii2=2`	设定换向区
`WAITP(Z)`	允许的摆动，用于Z轴

2. 运动同步动作

```
WHENEVER $AA_IM[Z]<$SA_OSCILL_REVERSE_POS2[Z]-ii2 DO ->
-> $AA_OVR[X]=0 $AC_MARKER[0]=0
```

如果	当前的、在MKS中的摆动轴Z的位置
小于	换向区2的开始，
则	设定横向进给轴的轴向倍率到0%。
并且	标记器（带变址0）设定到值0。

```
WHENEVER $AA_IM[Z]>=$SA_OSCILL_REVERSE_POS2[Z] DO $AA_OVR[Z]=0
```

如果	当前的、在MKS中的摆动轴Z的位置
大于小于	换向位置2，
则	设定摆动轴Z的轴向倍率到0%。

```
WHENEVER $AA_DTEPW[X]==0 DO $AC_MARKER[0]=1
```

如果	分度横向进给的剩余行程
相同于	零，
则	标记器（带变址0）设定到值1。

```
WHENEVER $AC_MARKER[0]==1 DO $AA_OVR[X]=0 $AA_OVR[Z]=100
```

如果	标记器带变址0
相同于	1，
则	设定横向进给轴X的轴向倍率到0%，由此防止一个过早的横向进给（摆动轴Z还没有再次离开换向区2，但是横向进给轴X已经用于一个新的横向进给）。 设定摆动轴Z的轴向倍率到100%（由此取消第二个同步动作）。

->必须在一个程序段中编程。

3. 启动摆动

`OSCILL[Z]=(X) POSP[X]=(5,1,1)`	启动轴轴X作为横向进给轴分配到摆动轴Z。 轴X应该按照每步为1运行到终点位置5。
`M30`	程序结束

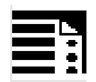
■

用于记录

冲裁和步冲

12.1 激活, 非激活

12.1.1 语言指令, SPOF, SON, PON, SONS, PONS, PDELAYON/OF



编程

```
PDELAYON
PON G... X... Y... Z...
PONS G... X... Y... Z...
PDELAYOF
SON G... X... Y... Z...
SONS G... X... Y... Z...
SPOF
PUNCHACC(Smin, Amin, Smax, Amax)
```

参数说明

PON	冲压开
PONS	冲压带预应力开
SON	步冲开
SONS	步冲带预应力开
SPOF	冲压，步冲关
PDELAYON	冲压带延迟开
PDELAYOF	冲压带延迟关
PUNCHACC	位移相关的加速 PUNCHACC (S_{min}, A_{min}, S_{max}, A_{max})
• "S_{min}"	最小的孔距
• "S_{max}"	最大的孔距
• "A_{min}"	开始加速度 A_{min} 可能大于 A_{max}
• "A_{max}"	结束加速度 A_{max} 可能小于 A_{min}



功能

冲压和步冲激活/非激活, PON/SON

使用PON和SON激活冲压或者步冲功能。SPOF结束所有的冲压和步冲专用的功能。

模态有效的指令PON和SON相互排斥，也就是说PON非激活SON，反之亦然。

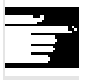
冲压和步冲，带预应力，PONS/SONS

功能SONS和PONS同样打开冲压功能或者步冲功能。
与SON/PON相反 － 插补平面上的冲程控制 －
在这些功能中，在伺服级面上对冲程释放进行信号控制。
由此您可以使用更高的冲程频率，以及更高的冲压功率。

在预应力信号计算期间所有导致步冲轴或者冲压轴位置改变的功能被闭锁。
举例:手轮运行，通过PLC改变框架，测量功能。

不可以同时在几个通道中进行带预应力的冲压和步冲。
PONS或者SONS仅可以在各自的通道中激活。

自软件版本**SW7.1**起，如果同时在几个通道中激活PONS或者SONS，则马上会引起报警2200“通道%1在几个通道中不可以进行快速冲压/步冲”。



POSN和SONS的其它方面的功能如同PON和SON。

位移相关的加速 PUNCHACC

语言指令PUNCHACC(S_{min}, A_{min}, S_{max}, A_{max})
确定一个加速度曲线，它根据孔距(S)定义不同的加速度(A)。举例 PUCHACC(2, 50, 10, 100)

2毫米以下的孔距：
使用50%的最大加速度的加速度运行。

孔距在2毫米到10毫米之间：
该加速度与距离成正比提高到100%。

孔距大于10毫米：
以100%的加速度运行。

带延迟的冲压

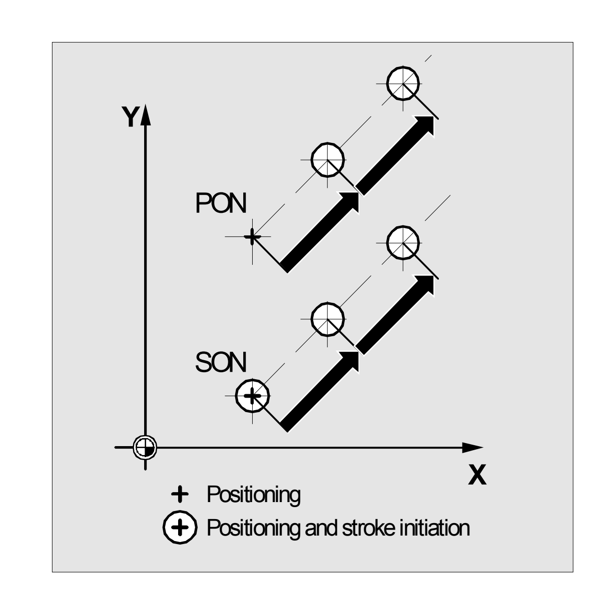
PDELAYON使冲压冲程延迟输出。模态有效的指令有预置的功能，并且通常位于PON之前。
在PDELAYOF之后又恢复到正常冲压。



冲程释放

释放第一个冲程

在步冲和冲压时，在激活该功能后释放第一个冲程在时间上不同。



PON/PONS:

- 所有的冲程 －
  即使在激活之后第一个程序段的冲程 －
  在程序段结束处发生。

SON/SONS:

- 在激活步冲之后，在程序段开始处已经发生第一个冲程。
- 所有的其它冲程在程序段结束处释放。

立即进行冲压和步冲

只有当程序段中包含冲压轴或者步冲轴（有效平面的轴）的运行信息时，才释放一个冲程。
为了要在相同的地点释放一个冲程，用运行位移0编程一个冲压轴/步冲轴。

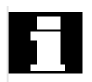


其它说明

用可旋转的刀具工作

为了可以在编程的轨迹处使用可旋转的刀具，请使用切线控制装置。

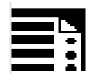
12.1.2 使用M指令



在使用宏指令技术时，
您可以不用语言指令而是使用M指令：

DEFINE M22 AS SON	步冲开
DEFINE M122 AS SONS	步冲带预应力开
DEFINE M25 AS PON	冲压开
DEFINE M125 AS PONS	冲压带预应力开
DEFINE M26 AS PDELAYON	冲压带延迟开
DEFINE M20 AS SPOF	冲压，步冲关
DEFINE M23 AS SPOF	冲压，步冲关

12.2 自动划分位移



编程

```
SPP=
SPN=
```



说明

SPP	分度距离的大小 (最大冲程距离); 模态方式有效
SPN	每个程序段分度距离的个数；程序段方式有效



功能

分度距离的细分

在有效的冲压时 或者步冲
时，不仅SPP而且SPN会产生一个划分，使给轨迹轴
编程的总的运行距离划分为一定量的相同长度的分度
距离 (等距离位移划分)。 在内部每个分
度距离相当于一个程序段。

冲程数

在冲压时第一个冲程发生在第一个分度距离的终点处，
相反，在步冲时则在第一个分度距离的起始处。
对于总的运行位移，因此就有以下数量：
冲压：
　冲程数＝分度距离数

步冲：
　冲程数＝分度距离数+1

辅助功能

辅助功能 在第一个所产生的程序段处执行。

12.2.1 在轨迹轴时的位移划分



工作流程

分度距离SPP的长度

使用SPP说明最大的冲程距离和分度距离的最大长度，按照该分度距离对总的运行距离进行划分。

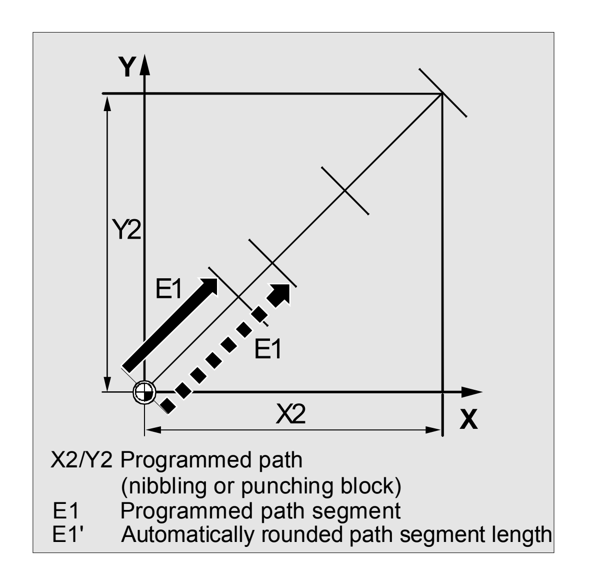
通过SPOF或者SPP=0关断该指令。

举例:

N10 G1 SON X0 Y0
N20 **SPP=2** X10

10毫米的总的运行距离划分为5个分度距离，每段2毫米（SPP=2）。

用SPP划分位移总是进行等距离地划分：所有的分度距离长度相同。
也就是说，只有当总的运行距离与SPP值的商为整数时，编程的分度距离大小（SPP值）才有效。
如果不是这种情况，则分度距离的大小在内部减少，从而产生一个整数的商。



X2/Y2 Programmed path (nibbling or punching block)
E1 Programmed path segment
E1' Automatically rounded path segment length

举例:

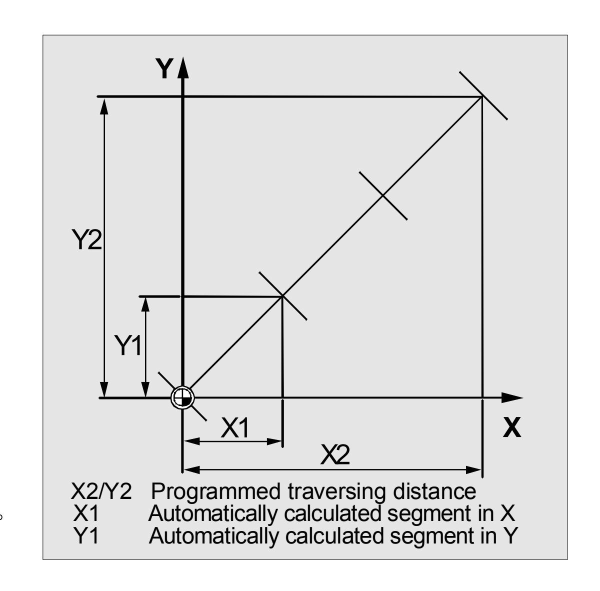
N10 G1 G91 SON X10 Y10
N20 SPP=3.5 X15 Y15

如果总的运行距离为15毫米，并且分度距离的一个长度为3.5毫米，则不会产生一个整数商（4.28）。
因此降低SPP值，直至下一个可能的整数商。在这种情况下产生一个3毫米的分度距离长度。

分度距离SPN的个数

使用SPN您可以定义分度距离的个数，它应该由总的运行距离产生。分度距离的长度会自动计算出来。

因为SPN为程序段方式生效，所以在事先必须激活冲压或者步冲，使用PON 或者 SON 。



SPP和SPN在同一个程序段中

如果您在一个程序段中不仅编程了分度距离长度（SPP），而且也编程了分度距离(SPN）个数，则SPN适用于该程序段，SPP适用于所有其它的程序段。

如果SPP已经在SPN之前激活，则它在SPN程序段之后再次生效。



其它说明

如果在控制装置中可以使用冲压/步冲，则使用SPN或者SPP编程的自动位移划分 可以激活，而与其工艺无关。

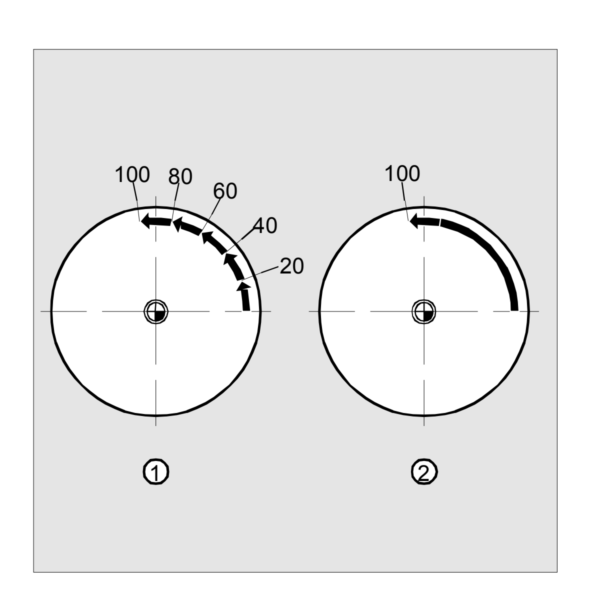
12.2.2 在单个轴时的位移划分

如果除了轨迹轴之外也有单个轴作为冲压－步冲－轴定义，则它们可能也会进行自动位移划分。

SPP时单个轴的特性

分度距离(SPP)的编程长度与轨迹轴相关。
因此在一个除了单个轴运动 和SPP
值之外没有编程轨迹轴的程序段中，SPP值被拒绝。

如果不仅编程了单个轴，而且在程序段中也编程了轨迹轴，则单个轴的特性取决于相应机床数据的设定。



1. 缺省设定
单个轴的位移均匀地分布到用SPP产生的中间程序段中。

举例:
N10 G1 SON X10 A0
N20 SPP=3 X25 A100

通过3毫米的冲程距离，在X轴（轨迹轴）15毫米的总的运行距离中生成5个程序段。
因此A轴在每个程序段中转动20°。

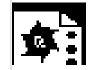
2. 没有位移划分的单个轴
单个轴在第一个所产生的程序段中运行总位移。

3. 不同的位移划分
单个轴的特性与轨迹轴的插补相关：

- 圆弧插补： 位移划分
- 线性插补： 没有位移划分

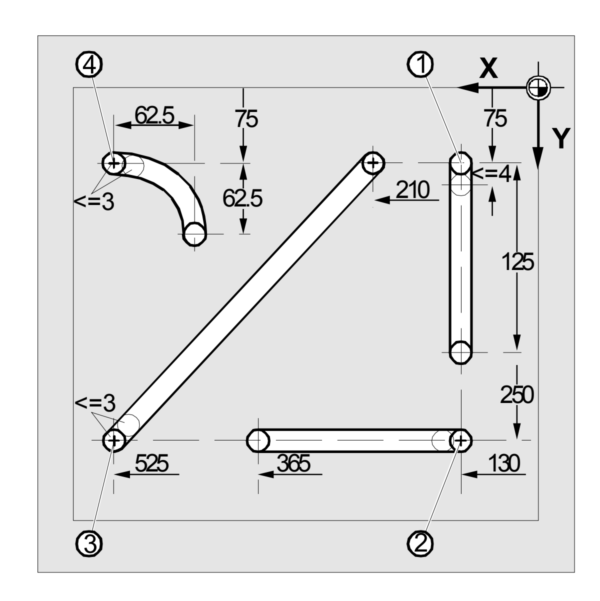
SPN时的特性
如果不是同时编程一个轨迹轴，则编程的分度距离的个数也适用。
前提条件：单个轴作为冲压－步冲－轴定义。

12.2.3 编程举例



编程举例 1

编程的步冲距离应该自动地分为相同大小的分度距离。



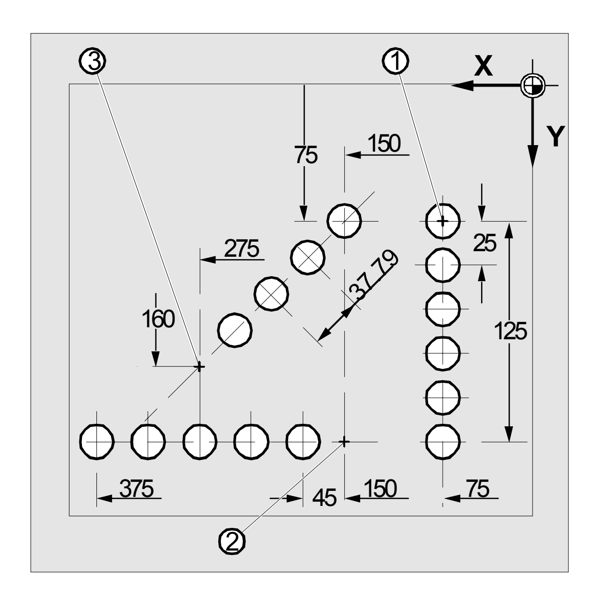
程序分段

N100 G90 X130 Y75 F60 SPOF	定位到起始点1
N110 G91 Y125 SPP=4 SON	步冲开；用于自动位移划分的最大的分度距离长度： 4 mm
N120 G90 Y250 SPOF	步冲关；定位到起始点2
N130 X365 SON	步冲开；用于自动位移划分的最大的分度距离长度： 4 mm
N140 X525 SPOF	步冲关；定位到起始点3
N150 X210 Y75 SPP=3 SON	步冲开；用于自动位移划分的最大的分度距离长度： 3 mm
N160 X525 SPOF	步冲关；定位到起始点4
N170 G02 X-62.5 Y62.5 I J62.5 SPP=3 SON	步冲开；用于自动位移划分的最大的分度距离长度： 3 mm
N180 G00 G90 Y300 SPOF	步冲关



编程举例 2

对于各个孔排列，应进行一次自动位移划分。划分时每次说明最大的分度距离长度（SPP值）。



程序分段

N100 G90 X75 Y75 F60 PON	定位到起始点1；冲压开；冲压单个孔
N110 G91 Y125 SPP=25	自动划分位移时最大的分度距离长度：25 mm
N120 G90 X150 SPOF	冲压关；定位到起始点2
N130 X375 SPP=45 PON	冲压开；用于自动位移划分的最大的分度距离长度： 45 mm
N140 X275 Y160 SPOF	冲压关；定位到起始点3
N150 X150 Y75 SPP=40 PON	冲压开；不是使用编程的40毫米的分度距离长度，而是使用计算的分度距离长度37,79 mm 。
N160 G00 Y300 SPOF	冲压关；定位

■

用于记录

其它功能

13.1 轴功能 AXNAME, SPI, ISAXIS, AXSTRING (自软件版本 SW 6起)



编程

```
AXNAME("PLANACHSE")
AX[AXNAME("String")]
AXSTRING ( SPI(n) )
SPI(n)
ISAXIS(几何轴号)
```

命令解释

AXNAME	输入端字符串转换为轴名称； 输入端字符串必须包含有效的轴名称。
SPI	主轴号转换为轴名称；传送参数中必须包含一个有效的主轴号。
n	主轴号
AXSTRING	至软件版本SW5为止，主轴所分配的轴的轴变址作为主轴号输出。 自软件版本SW6起，该字符串连同所分配的主轴号输出。
AX	变量轴名称
ISAXIS	检查是否有所说明的几何轴。



功能

使用AXNAME，比如在编制通用的循环时，当轴名称并不知晓时（也可参见章节13.10“字符串功能”）。

使用SPI，当轴功能用于一个主轴时，比如同步主轴。

在通用循环中使用ISAXIS，保证具有一个确定的几何轴，并且后面的$P_AXNX-调用不会被出错停止。

（自软件版本SW6起）

扩展 SPI(n)：

轴功能SPI(n)现在也用于读写框架部件。

由此可以写框架，比如用句法

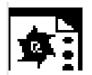
$P_PFRAME[SPI(1),TR]=2.22。

通过地址AX[SPI(1)] = <轴位置>

附加编程轴位置，可以运行一个轴。

故障消除， 在AXSTRING(SPI(n))时
至软件版本SW5为止，在编程AXSTRING(SPI(n))
时，主轴所分配的轴的轴变址作为主轴号输出。
举例:
主轴1分配到第5轴
($MA_SPIND_ASSIGN_TO_MACHAX[AX5]=1),
AXSTRING(SPI(1))提供错误的字符串"S4"。

自软件版本SW6起，使用AXSTRING[SPI(n)
]输出字符串"Sn"。
举例:
AXSTRING(SPI(2)) 提供字符串 "S2"。



编程举例

作为端面轴定义的轴应该运动。

OVRA[AXNAME("Planachse")]=10	端面轴
AX[AXNAME("Planachse")]=50.2	端面轴的终点位置
OVRA[SPI(1)]=70	主轴1的倍率
IF ISAXIS(1)==FALSE GOTOF WEITER	有横坐标吗？
AX[$P_AXN1]=100	横坐标运行
WEITER:	



13.2 功能调用 ISVAR ( ) (自软件版本 SW 6.3起)

编程

```
ISVAR ("变量名称")
ISVAR (名称, [值, 值])
```

命令解释

变量名称	字符串类型的传送参数既可以没有尺寸，也可以单尺寸或者二维尺寸。
名称	名称，带一个NC知晓的变量，带或者不带行变址作为机床数据、设定数据、系统变量或者通用的变量。
值	BOOL类型的功能值

结构

传送参数可以按如下方式构成：

1. 无尺寸的变量：
   名称
2. 单个尺寸变量，不带行变址：
   名称[ ]
3. 单个尺寸变量，带行变址：
   名称[值]
4. 二维尺寸变量，不带行变址：
   名称[,]
5. 二维尺寸变量，带行变址：
   名称[值，值]



功能

ISVAR指令是一个具有NC语言含义的功能，带一个：

- 功能值，类型　BOOL
- 传送参数，类型　STRING

ISVAR指令提供TRUE（真），
如果传送参数包含一个NC知晓的变量（机床数据，
设定数据，系统变量，通用变量如GUD`s)。

检查

根据传送参数进行以下的检查：

- 是否有名称
- 是否与一个尺寸或者二维的数组有关
- 是否允许一个行变址

只有当所有这些检查为正时，才送回TRUE（真）。如果仅有一个检查不能满足，或者仅出现一个语法错误，则它用FALSE（假）应答。轴向的变量作为轴名称的变址接收，但是没有进一步检查。

举例：

```
DEF INT VAR1
DEF BOOL IS_VAR=FALSE                    ；传送参数是通用变量
N10 IS_VAR=ISVAR(“VAR1“)                 ；IS_VAR 在这种情况下是 TRUE（真）
DEF REAL VARARRAY[10,10]
DEF BOOL IS_VAR=FALSE                    ；不同的语法变量
N20 IS_VAR=ISVAR(“VARARRAY[,]“)          ；IS_VAR 是 TRUE（真），带二维的行矩阵
N30 IS_VAR=ISVAR(“VARARRAY“)             ；IS_VAR 是TRUE（真）, 变量存在
N40 IS_VAR=ISVAR                         ；IS_VAR 是 FALSE（假）, 行变址不允许
    (“VARARRAY[8,11]“)
N50 IS_VAR=ISVAR(“VARARRAY[8,8“)         ；IS_VAR 是 FALSE（假）, 句法错误用于缺少的 “]“
N60 IS_VAR=ISVAR(“VARARRAY[,8]“)         ；IS_VAR 是 TRUE（真）, 行变址允许
N70 IS_VAR=ISVAR(“VARARRAY[8,]“)         ；IS_VAR 是 TRUE（真）

DEF BOOL IS_VAR=FALSE                    ；传送参数是一个机床数据
N100 IS_VAR=ISVAR                        ；IS_VAR 是 TRUE（真）
    (“$MC_GCODE_RESET_VALUES[1]“

DEF BOOL IS_VAR=FALSE                    ；传送参数是一个系统变量
N10 IS_VAR=ISVAR(“$P_EP“)                ；IS_VAR 在这种情况下是 TRUE（真）
N10 IS_VAR=ISVAR(“$P_EP[X]“)             ；IS_VAR 在这种情况下是 TRUE（真）
```



13.3 学习补偿特征曲线： QECLRNON, QECLRNOF

命令解释

QECLRNON (Achse.1,…4)	打开功能“学习象限缺陷补偿”
QECLRNOF	关闭功能“学习象限缺陷补偿”

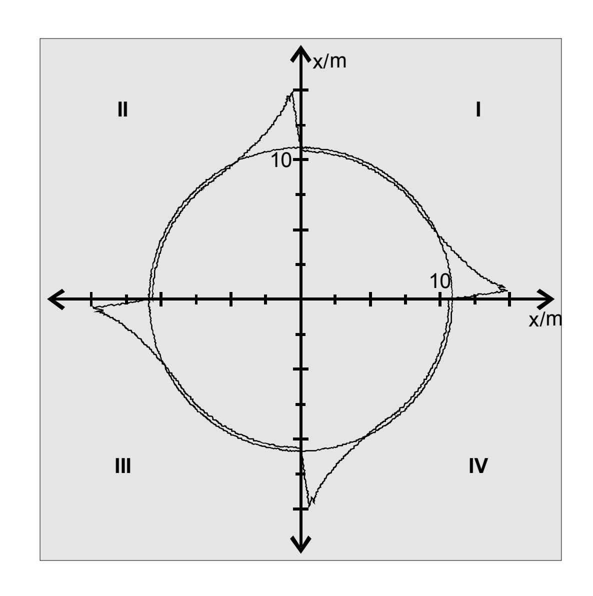


功能

象限缺陷补偿 (QFK)
降低轮廓误差，这种轮廓误差在运行方向转换时由于机械的非线性（比如摩擦，间隙) 或者扭转 而产生。

优化的补偿数据可以在学习期间由控制系统根据一个神经原网络进行适配，并且可以自动计算出补偿特征曲线。

这种学习可以最多4个轴同时进行。





工作流程

学习所要求的轴的位移运动可以利用NC程序产生。
这里学习运行以一个学习循环的形式出现。

第一次学习
在开机调试第一次学习时，在PLC基本程序的磁盘中包含样本NC程序，用于学习运行的学习，以及用于学习QFK系统变量的占用：

QECLRN.SPF	学习循环
QECDAT.MPF	样本NC程序，用于系统变量的占用以及学习循环的参数设定。
QECTEST.MPF	样本NC程序，用于圆弧测试

补充学习

重新优化已经学习的特征曲线可以用“补充学习”进行。在到目前为止用户存储器中的数据上进行。

为了补充学习，您可以把样本NC程序适配于您的要求。

有时学习循环 (比如 QECLRN.SPF)
的参数必须进行修改，用于“补充学习”

- 设置"学习方式" = 1
- 有时减少“学习过程次数”
- 有时激活“分段方式学习”，并确定相应的范围极限

激活学习过程: QECLRNON

自身的学习过程在NC程序中用指令QECLRNON激活，说明轴。

```
QECLRNON (X1, Y1, Z1, Q)
```

只有当该指令有效时，才改变此特征曲线。

关断学习: QECLRNOF

在结束所要求轴的学习运行之后，学习过程用QECLRNOF关断，同时用于所有的轴。

13.4 同步主轴

编程

```
COUPDEF (FS,LS,ÜFS,ÜLS,Satzverh.,Koppel)
COUPDEL (FS,LS)
COUPRES (FS,LS)
COUPON (FS,LS,PSFS)
COUPOF (FS,LS,POSFS,POSLS)
WAITC (FS,Satzverh.,LS,Satzverh.)
```

命令解释

COUPDEF	编制/修改用户定义的耦合
COUPON	打开耦合
COUPOF	关断耦合
COUPRES	复位耦合参数
COUPDEL	删除用户定义的耦合
WAITC	等待同步运行条件

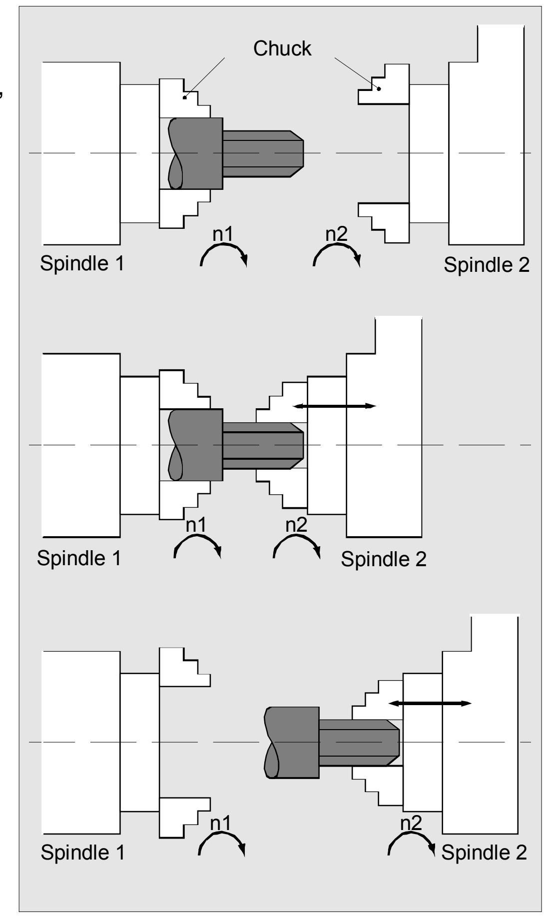
参数说明

FS, LS	跟随主轴和引导主轴的名称；用主轴号说明 比如S2
$Ü_{FS}$, $Ü_{LS}$	跟随主轴和引导主轴的传动参数 预设置＝1.0；分母优化说明
程序段关系：:	；进行程序段转换：
• "NOC"	立即（预设置）
• "FINE"	在"精同步运行"
• "COARSE"	在 "粗同步运行"
• "IPOSTOP"	在 IPOSTOP时(也就是说在给定值一侧的同步运行之后)
耦合	耦合方式:FS和LS之间的耦合
• "DV"	给定值耦合 (预设置)
• "AV"	实际值耦合
PS_{FS}	在引导主轴和跟随主轴之间的角度偏差
POS_{FS}, POS_{LS}	跟随主轴和引导主轴的关断位置



功能

在同步运行中有一个引导主轴(LS) 和一个跟随主轴(FS),也就是所谓的**同步主轴副** 。
在耦合有效时（同步运行）主轴跟随引导主轴的运动，根据所确定的功能关系。

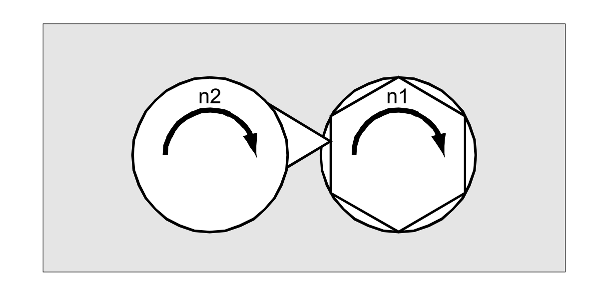


该功能在车床中提供一种快速传送工件的可能性，此时主轴1运行，变换到主轴2，比如至最终加工，从而避免重新夹装的辅助时间。

工件传送可以进行：

- 转速同步 ($n_{FS} = n_{LS}$)
- 位置同步 ($\varphi_{FS} = \varphi_{LS}$)
- 位置同步，带角度偏差
  ($\varphi_{FS} = \varphi_{LS} + \Delta\varphi$)

因此主主轴和“刀具主轴”之间的传动比$k_Ü$的给定是多棱边加工的前提条件（多边形车削）。





同步主轴副可以利用通道专用的机床数据固定设计，用于每个机床，或者通过CNC零件程序根据应用场合定义。
每个NC通道可以同时运行最多2个同步主轴副。



工作流程

确定同步主轴副：方法

固定设计的耦合：
引导主轴和跟随主轴通过机床数据确定。
在这种耦合中，用于LS和FS固定确定的加工轴不可以由NC零件程序修改。但是在NC零件程序中仍然可以使用COUPDEF进行耦合的参数设定
(前提条件:确定没有写保护)。

用户定义的耦合：
使用语言指令COUPDEF可以在零件程序中重新设定耦合并进行修改。如果要求定义一个新的耦合关系，则有时必须事先用t COUPDEL 删除一个
已经存在的、用户定义的耦合。

COUPDEF定义新的耦合

在下面说明预定义的子程序参数设定：
COUPDEF (FS,LS,$Ü_{FS}$,$Ü_{LS}$,程序段关系,耦合)

跟随主轴和引导主轴：FS 和 LS

使用轴名称FS和LS明确确定耦合。
它们必须在每个COUP指令中编程。其它的耦合参数只有在要求改变时（模态有效）才进行编程。

举例:
```
N...  COUPDEF(S2,S1,ÜFS,ÜLS)
```
意义：
S2 = 跟随主轴,, S1 = 引导主轴

定位跟随主轴：方法

当同步主轴耦合接通后，跟随主轴也与引导主轴引起的运动无关，在±180°范围内定位。

SPOS定位

跟随主轴可以使用 SPOS=… 插补。

有关SPOS的详细信息参见编程说明基础部分。

举例:

```
N30 SPOS[2]=IC(-90)
```

FA, ACC, OVRA :

速度，加速度

使用 FA [SPI (Sn)] 或者 FA[Sn], ACC[SPI(Sn)] 或者 ACC [Sn] 和 OVRA [SPI(n)] 或者 OVRA[Sn] 可以编程定位速度和加速度值，用于跟随主轴（参见编程说明基础部分）。"n" 指主轴号1...n。

可编程的程序段转换WAITC

使用WAITC可以确定程序段转换特性用于程序继续，比如在改变耦合参数或者定位过程之后，带有不同的同步运行条件 (粗, 精, IPOSTOP)。

由此，延迟新程序段的换入，直至达到同步运行条件，从而可以更快地执行同步运行。

如果没有说明同步运行条件，则给各个耦合编程/设计的程序段转换特性适用。

举例：

```
N200 WAITC
```

等待同步运行条件，用于所有有效的跟随主轴，没有说明同步运行条件。

```
N300 WAITC(S2,"FINE",S4,"COARSE")
```

等待所说明的同步运行条件“粗”，用于跟随主轴S2和S4。

传动比$k_Ü$

传动比通过说明FS（分子）和LS（分母）值规定。

方法：

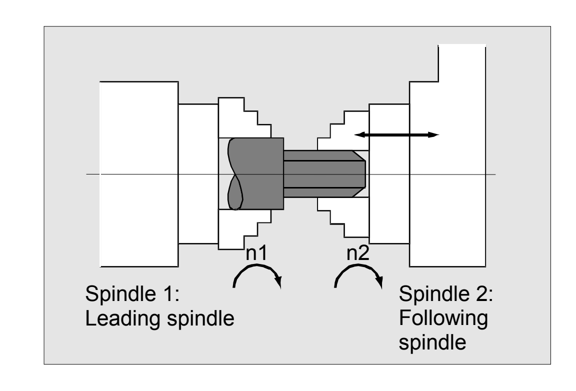
- 跟随主轴和引导主轴以相同的转速旋转
  ($n_{FS} = n_{LS}$; $k_Ü$ 正)
- 在LS和FS之间同方向或者反方向($k_Ü$ 负)运行
- 跟随主轴和引导主轴以不同的转速旋转
  ($n_{FS} = k_Ü \cdot n_{LS}$; $k_Ü \neq 1$)
  应用：多边形车削

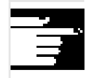
举例:

```
N…  COUPDEF(S2, S1, 1.0, 4.0)
```

意义：

跟随主轴 S2 和引导主轴 S1 以传动比 0.25旋转。





- 至少分子必须编程。如果没有说明分母，则分母设定为“1”。
- 在接通耦合期间也可以在运动中改变传动比。

程序段转换特性

因此在定义耦合时可以在下面的方法之间进行选择，何时进行程序段转换：

"**NO**C"	立即（预设置）
"**FI**NE"	在"精同步运行"
"**CO**ARSE"	在 "粗同步运行"
"**IP**OSTOP"	在IPOSTOP时 （也就是说给定值一侧的同步运行之后）。

为了说明程序段转换特性，写粗写体字母就足够了。

程序段转换特性模态方式有效！

耦合方式

"DV"	在FS和LS之间额定值耦合（预设置）。
"AV"	FS和LS之间的实际值耦合

耦合方式模态有效。



注意

耦合方式仅在关断耦合时才可以修改。

接通同步运行

- 在LS和FS之间以任意角度关系
  最快可能的耦合接通：

```
N … COUPON (S2, S1)
```

- 开通角度偏置POS_{FS}
  用于成型工件的位置同步耦合。
  POS_{FS}与引导主轴在正方向的0°位置相关。

  值范围 POS_{FS}: 0°… 359,999°:

```
COUPON (S2,S1,30)
```

以这种方式您也可以在有效的耦合时修改角度偏差。

关断同步运行 COUPOF

下面的变量可能：

- 最大可能快速地关断耦合，程序段转换立即使能：

```
COUPOF (S2,S1)
```

- 在超出关断位置后； 程序段转换在超出关断位置
  POS_{FS} 和 POS_{LS} 之后使能。

  值范围 0°… 359,999°:

```
COUPOF (S2,S1,150)
COUPOF (S2,S1,150,30)
```

关断耦合，停止跟随主轴COUPOFS
（自软件版本SW6.4起）
可以有两个变量：

- 最大可能快速地关断耦合和停止，没有位置说明，程序段转换立即使能：

```
COUPOFS (S2,S1)
```

- 越过编程的跟随轴－关断位置，它以机床坐标系为基准，程序段在越过关断位置POS_{FS}之后才使能。

```
COUPOFS (S2,S1,POSFS)
```

值范围 0°… 359,999°:

删除耦合 COUPDEL (至软件版本 SW 6.3)
如果要求定义了一个新的耦合关系，并且已经确定了所有可能自由配置的耦合（1或者2），则必须删除当前用户定义的同步主轴耦合。

```
N … COUPON (S2,S1)
```

SPI(2) = 跟随主轴, SPI(1) = 引导主轴


只有当该耦合取消选择之后 (COUPOF)才可以删除。一个固定设计的耦合不可以用COUPDEL删除（至软件版本6.3）。


删除耦合 COUPDEL (自软件版本 SW 6.4起)
在一个有效的同步主轴耦合中，现在也可以关断耦合，并删除耦合数据。跟随主轴接收最后的转速，符合COUPOF(FS,LS)的当前特性。

复位耦合参数 COUPRES

使用语言指令 "COUPRES"，

- 使机床数据中和设定数据中存储的参数激活（固定设计的耦合）
- 使预设置激活（用户定义的耦合）

在此，用COUPDEF编程的参数（包括传动比）丢失。

```
N … COUPRES (S2,S1)
```

S2 = 跟随主轴, S1 = 引导主轴

系统变量

跟随主轴当前的耦合状态

对于跟随主轴，可以读出在NC零件程序中带下面轴向系统变量的、当前的耦合状态：

```
$AA_COUP_ACT[FS]
```

FS = 跟随主轴轴名称，带主轴号，比如 S2。

所读出的值对于跟随主轴有以下的含义：
0: 没有耦合有效
4: 同步主轴耦合有效

当前的角度偏差

可以读出在NC零件程序中带下面轴向系统变量的、当前给定值一侧的位置偏差（FS对LS）：

```
$AA_COUP_OFFS[S2]
```

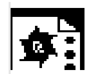
实际值一侧的位置偏差可以由以下参数读出：

```
$VA_COUP_OFFS[S2]
```

FS = 跟随主轴轴名称，带主轴号，比如 S2。



在开启的耦合和跟随运行中取消调节器使能，并且在重新给予一个调节器使能之后，出现一个新的的位置偏置，它与原来编程的值不同。在这种情况下可以读出修改的位置偏差，该修改的位置偏差有时可以在NC零件程序中进行校准。



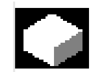
编程举例

带引导主轴和跟随主轴的加工。

	；引导主轴 = 主主轴 = 主轴 1
	；跟随主轴 = 主轴 2
N05 M3 S3000 M2=4 S2=500	；引导主轴旋转3000/分钟，跟随主轴500/分钟
N10 COUPDEF (S2, S1, 1, 1, "NOC", "Dv")	；定义耦合；也可以设计
…	
N70 SPCON	；引导主轴接收到位置调节回路（给定值回路）
N75 SPCON(2)	；跟随主轴接收到位置调节回路
N80 COUPON (S2, S1, 45)	；快速耦合到偏差位置＝45度
…	
N200 FA [S2] = 100	；定位速度＝100度/分钟
N205 SPOS[2] = IC(-90)	；在负方向重叠90度运行
N210 WAITC(S2, "Fine")	；等待同步运行“精”
N212 G1 X… Y… F…	；加工
…	
N215 SPOS[2] = IC(180)	；在负方向重叠90度运行
N220 G4 S50	；停留时间＝50转主主轴
N225 FA [S2] = 0	；激活设计的速度（MD）
N230 SPOS[2] = IC (-7200)	；20转在负方向以设计的速度
…	
N350 COUPOF (S2, S1)	；快速去除耦合，S=S2=3000
N355 SPOSA[2] = 0	；在零度时停止FS
N360 G0 X0 Y0	
N365 WAITS(2)	；等待主轴2
N370 M5	；停止FS
N375 M30	



13.5 EG:电子齿轮 (自软件版本 SW 5起)



序言

利用功能“电子齿轮”可以按照线性运动程序段控制一个跟随轴的运动，最多与5个引导轴相关。每个引导轴可以通过耦合系数定义引导轴和跟随轴之间的关系。

通过使单个引导轴运动分量乘以其耦合系数，由加法构成所计算的跟随轴运动分量。

在激活一个EG轴组合时，可以在一个定义的位置开始跟随轴的同步。

一个齿轮传动可以由零件程序：

- 定义，
- 接通，
- 关断，
- 删除。

跟随轴运动可以从以下值中导出（可选择）：

- 引导轴给定值，以及
- 引导轴实际值

自软件版本6起，作为扩展，也可以通过曲线表（参见第9章）实现引导轴和跟随轴之间的非线性关系。

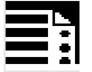
电子齿轮可以级联，也就是说一个电子齿轮的跟随轴可以是下一个电子齿轮的引导轴。

13.5.1 定义电子齿轮： EGDEF



功能

一个EG轴关联可以通过说明跟随轴，和至少一个、至多五个引导轴与其耦合类型确定：
EGDEF(跟随轴, 引导轴1, 耦合类型1, 引导轴2, 耦合类型2,..)



说明

跟随轴	受引导轴影响的轴
引导轴1, ...引导轴5	影响跟随轴的轴
耦合类型1, ...耦合类型5	跟随轴受相应引导轴的： 0: 实际值 1: 给定值 影响



编程

```
EGDEF(C, B,1, Z, 1, Y, 1)
```
B, Z, Y 通过给定值影响 C

耦合类型并不是所有的引导轴都相同，因此每个引导轴可以单个说明。
在定义EG耦合关联时，用零预置耦合系数。
定义一个EG轴关联的前提条件：
对于跟随轴，不允许定义轴耦合（有时必须事先用EGDEL删除一个已经存在的轴耦合）。

说明

EGDEF删除进刀停止。带EGDEF的齿轮定义也可以无改变地使用，
如果在软件版本**SW6**以后的系统中，一个或者多个引导轴通过曲线表对跟随轴影响。

13.5.2 接通电子齿轮

接通指令有3种型式：

- **型式1**：

EG轴关联可以没有同步，选择使用以下参数接通：

```
EGON(FA, "程序段转换方式", LA1, Z1, N1,
LA2 , Z2, N2,..LA5, Z5, N5.)
```

说明

FA	跟随轴
程序段转换方式	可以使用以下的方式： ”NOC” 程序段转换立即 ”FINE” 程序段转换，在”精同步运行” ”COARSE” 程序段转换，在“粗同步运行” ”IPOSTOP” 程序段转换，在给定值一侧的同步运行
LA1, ... LA5	引导轴
Z1, ... Z5	分子，用于耦合系数i
N1, ... N5	分母，用于耦合系数i 耦合系数 i = 分子 i / 分母 i

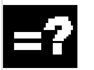
仅允许编程事先用EGDEF规定参数的引导轴。至少必须编程一个引导轴。

接通时刻的引导轴和跟随轴位置作为“同步位置”存储。可以使用系统变量 $AA_EG_SYN 读出“同步位置”。

- **型式2**：

EG轴关联可以带同步、选择使用以下参数接通：

```
EGONSYN(FA, "程序段转换方式", SynPosFA,[, LAi, SynPosLAi, Zi, Ni])
```



说明

FA	跟随轴	
程序段转换方式	可以使用以下的方式：	
	”NOC”	程序段转换立即
	”FINE”	程序段转换，在 ”精同步运行”
	”COARSE”	程序段转换，在 “粗同步运行”
	”IPOSTOP”	程序段转换，在 给定值一侧的同步运行
[, LAi, SynPosLAi, Zi, Ni]	（不写方括号）最少1，最多5个跟随：	
LA1, ... LA5	引导轴	
SynPosLAi	用于第I个引导轴的同步位置	
Z1, ... Z5	分子，用于耦合系数i	
N1, ... N5	分母，用于耦合系数i 耦合系数 i = 分子 i / 分母 i	

- **型式3：**

EG轴关联可以带同步、选择接通。返回运行方式规定，用：

```
EGONSYNE(FA, "程序段转换方式", SynPosFA, 返回运行方式
[, LAi, SynPosLAi, Zi, Ni])
```

说明

	这些参数与变量2的相同：	
返回运行方式：	可以使用以下的方式：	
	"NTGT"	下一个齿槽 以时间最优方式返回
	"NTGP"	下一个齿槽 以位移最优方式返回
	"ACN"	回转轴在负方向 运行 绝对
	"ACP"	回转轴在正方向 运行 绝对
	"DCT"	时间最优方式 到达编程的 同步位置

	"DCP"	位移最优方式 到达编程的 同步位置

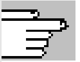
变量3仅对耦合到取模－引导轴的取模－跟随轴产生影响。以时间最优方式考虑跟随轴的速度极限。齿间距（度）由下面公式产生： 360 * Zi/Ni.
在调用时跟随轴如果停止，则位移优化方式时特性与时间优化方式时一样。在跟随轴已经运行时，用NTGP同步到下一个齿槽，而与跟随轴当前的速度无关。

在跟随轴已经运行时，用NTGT同步到下一个齿槽，而与跟随轴当前的速度无关。有时轴也会制动。

软件版本6

如果一个曲线表用于一个引导轴，则必须：

Ni	线性耦合耦合系数的分母设置为0。（分母0对于线性耦合是不允许的）。分母零对于系统来说是一个标记，即
Zi	待使用的曲线表应作为分数解释。带所说明的分数的曲线表必须在接通时刻已经定义。
LAi	引导轴的参数与通过耦合系数耦合时的引导轴参数一样（线性耦合）。



有关如何使用曲线表和级联电子齿轮及其同步可以参见：
/FB/ M 3 , 联动, 引导值耦合

仅允许编程事先用EGDEF规定参数的引导轴。

通过给跟随轴(SynPosFA)和引导轴(SynPosLA)编程“同步位置”，定义耦合关联作为同步使用的位置。
如果电子齿轮在接通时不在同步状态，则跟随轴运行到其定义的同步位置。
如果取模轴处于耦合关联状态，则其位置值取模降低。由此可以保证返回运行到最近可能的同步位置（所谓的相对同步：比如下一个齿槽）。如果跟随轴没有给出“使能跟随轴叠加”接口循环DB(30+轴号），DBX26位4，则不运行到同步位置。与此相反，程序在EGONSYN－程序段处停止，并且只要上面的信号设置，就给出自删除的报警16771。

13.5.3 关断电子齿轮

对于关闭一个有效的EG轴关联，有以下三种方法：

型式1：

```
EGOFS(跟随轴)
```

关断电子齿轮。跟随轴制动到停止。
此调用删除进刀停止。

型式2：

```
EGOFS(跟随轴, 引导轴 1, ...引导轴5)
```

通过设定该指令的参数可以选择消除单个引导轴对跟随轴运动的影响。

至少必须说明一个引导轴。所说明的引导轴对跟随轴的影响可以有目标的关断。
此调用删除进刀停止。
如果还有有效的引导轴，则跟随轴在此影响下继续运行。如果所有的引导轴影响都以这种方式关闭，则跟随轴被制动到停止。

型式3：

EGOFC(跟随主轴)

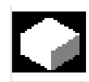
关断电子齿轮。跟随主轴以关闭时刻有效的转速/速度继续运行。
此调用删除进刀停止。



说明

该功能仅允许用于主轴。

13.5.4 删除一个电子齿轮的定义



根据前面的章节，在可以删除其定义之前，必须使EG轴关联处于关闭状态。

EGDEL(跟随轴)

轴关联的耦合定义被删除。
在到达同时激活的轴关联最大个数之前，又可以用EGDEF重新定义其它的轴关联。此调用删除进刀停止。

13.5.5 旋转进给 (G95)/电子齿轮 (SW 5.2)



自软件版本SW5开始，可以使用FPR()也可以说明一个电子齿轮的跟随轴，作为旋转进给时进给确定的轴。对于这种情况，以下的特性适用：

- 进给取决于电子齿轮跟随轴的给定速度。
- 给定速度可以由引导轴和取模－引导轴（不是轨迹轴）的速度和相应的耦合系数计算出来。
- 不考虑线性或非模态引导轴和跟随轴的叠加运动的速度分量。

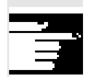
13.5.6 在Power On (上电)、RESET（复位）、 运行方式转换、搜索时的EG性能

在上电之后没有耦合有效。
在复位和运行方式转换之后有效的耦合仍保持。
在程序段搜索时，有关开关、删除、定义电子齿轮的指令不予执行和考虑，而是直接跳过。

13.5.7 电子齿轮的系统变量



利用电子齿轮的系统变量，零件程序可以求值一个EG轴关联的当前状态，有时并做出反应。



其它说明

您可以在附录中找到电子齿轮的系统变量。它们被标记，通过名称，以如下方式开始：

```
$AA_EG_ ...
```

或者

```
$VA_EG_ ...
```



13.6 扩展的停止和退回 (自软件版本 SW 5起)



功能

功能“扩展的停止和退回”
ESR(Extended Stopping and Retract)
提供一种可能性，即对可选择的故障源可以进行灵活的、保护工件的反应。

“扩展的停止和退回”可以使用以下的部分反应：

- "**扩展的停止**" (驱动自给, SW 5)
  是一种定义的、时间延迟的停止。
- "**退回**" (驱动自给)
  表明从加工平面“退回”到一个安全的返回位置。
  由此应该避免一个刀具和工件之间的冲撞危险。
- "**发生器运行**" (驱动自给)
  当中间回路的能量不足以安全退回时，可能会有一个发生器运行。在电网故障时，作为自身的驱动运行方式，可以使用一个必要的能量，用于驱动中间回路，它用于一个有秩序的“停止”和“退回”。

自软件版本**SW6**起，另外：

- **扩展的停止** (NC–控制)
  是一种NC控制的，已经定义的、时间延迟的并且保护轮廓的停止。
- **退回** (NC–控制)
  是一种NC控制的，由加工平面“退回”到一个安全的退回位置。由此可以避免在刀具和工件之间出现碰撞危险。齿轮加工时，可以理解为从当前加工的齿槽退回。

所有的反应可以相互无关地加以利用。



其它的信息参见
/FB/ M3, 轴耦合和 ESR。

13.6.1 驱动自给的反应



功能

驱动自给反应为轴向定义，也就是说每个驱动在激活的情况下自给执行它的停止/退回－要求。没有给出停止时或者退回时轴的插补耦合或者轨迹定义的耦合，轴的基准由时间控制运行。
在执行驱动自给的反应期间和之后，其驱动不再受NC使能或者NC运行指令控制。要求一次关机再开机。
报警“26110：释放驱动自给的停止/退回”对此说明。

发生器运行

发生器运行

- 配置：通过 MD 37500: **10**
- 使能：系统变量
  $AA_ESR_ENABLE
- 激活：取决于低于中间回路电压时驱动－机床数据的设定。

退回（驱动自给）

驱动自给的退回

- 配置：通过 MD 37500:**11**;
  时间参数和退回速度通过 MD进行，参见
  "举例：驱动自给反应的使用”，在本章结束处。
- 使能：系统变量
  $AA_ESR_ENABLE
- 释放：系统变量
  $AN_ESR_TRIGGER。

停止（驱动自给）

驱动自给的停止

- 配置：通过 MD 37500:12
  以及时间参数，通过MD；
- 使能 ($AA_ESR_ENABLE) 并且
- 启动：系统变量 $AN_ESR_TRIGGER。

13.6.2 NC控制的反应



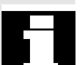
功能

退回

前提条件：

- 用 POLFMASK 或者POLFMLIN 选择的轴
- 用POLF定义的轴位置
- 用POLFA定义的单个轴的退回位置
  （自软件版本SW6.4起）
- 时间窗口，在
  MD 21380:ESR_DELAY_TIME1 和
  MD 21381: ESR_DELAY_TIME2
- 触发系统变量
  $AC_ESR_TRIGGER
  $AA_ESR_TRIGGER用于单个轴
- 相符的 ESR–反应
  MD 37500: ESR_REACTION = 21
- LFPOS来自模态的第46个G代码组

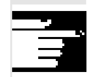
如果系统变量$AC_ESR_TRIGGER = 1设置，
并且在该通道中配置了一个退回轴
（也就是说MD37500:ESR_REACTION = **21**)
并且设置轴 $AA_ESR_ENABLE = 1，
则在该通道**LIFTFAST**中激活。



在零件程序中，必须编程退回位置POLF。
在用POLFA（轴，类型，值）单个轴退回时，必须编程值，并且遵守以下的条件：

- $AA_ESR_ENABLE = 1 设置。
- POLFA(轴)在触发时刻必须是单个轴。
- POLFA(类型) 或者 类型=1 或者 类型=2

对于退回运动必须已经设置使能信号，并且保持设置。



其它说明

用**LFPOS,POLF**配置的退刀运动（用**POLFMASK**或者**POLFMLIN**选择的轴）代替这些轴在零件程序中所确定的轨迹运动。

扩展的退回运动（也就是说通过$AC_ESR_TRIGGER释放的LIFTFAST/LFPOS）不可以中断，并且仅可以通过NOTAUS提前结束。对于退回运动，最多可以使用时间MD 21380:ESR_DELAY_TIME1和MD 21381:ESR_DELAY_TIME2之和。在该段时间结束之后，对于退回轴也引入快速制动，接着跟随运行。

在激活快速退刀时刻考虑有效的框架。

重要：

带旋转的框架通过**POLF**也影响退刀在哪一个方向进行。

NC控制的退回

- 配置：通过 MD 37500:**21** 以及
  2 个时间参数，通过 MD s.o.;
- 使能 ($AA_ESR_ENABLE) 并且
- 启动：系统变量 $AC_ESR_TRIGGER
  用于单个轴，带 $AA_ESR_TRIGGER

举例说明单个轴的退回

MD 37500: ESR_REACTION[AX1] = 21	；NC控制的退回
...	
$AA_ESR_ENABLE[AX1] = 1	
POLFA(AX1,1, 20.0)	；AX1分配轴向退回位置20.0（绝对）。
$AA_ESR_TRIGGER[AX1] = 1	；从这里开始退回运动。



编程

`POLF[geo |mach]= 值`	退回轴目标位置
`POLFA(轴, 类型, 值)`	单个轴退回位置 允许以下的字母简式：
`POLFA(轴, 类型)`	字母简式，用于单个轴退回
`POLFA(轴, 0/1/2)`	快速非激活或者激活
`POLFA(轴, 0, $AA_POLFA[轴])`	引起一次进刀停止
`POLFA(轴, 0)`	不引起进刀停止
`POLFMASK(achsname1, achsname2, ...)`	轴选择，用于退回 轴，没有关联
`POLFMLIN(轴名1, 轴名2, ...)`	轴选择，用于退回 轴，带线性关联



警告

如果在使用字母简式POLFA时仅改变类型，则用户必须保证退回位置或者退回位移必须包含一个有意义的值。特别是在上电以后必须重新设置退回位置和退回位移。

命令解释

geo | mach	几何轴或者 退回的通道轴/加工轴。
轴	所适用的单个轴的轴名称（自软件版本SW6.4）
类型	单个轴的位置值（自软件版本SW6.4起），类型： 0 位置值使无效 1 位置值为绝对的 2 位置值为增量的（距离）
值	退回位置，WKS适用于几何轴，MKS适用于其它轴。如果几何轴和通道轴/加工轴名称相同，则在工件坐标系中退回。 允许增量式编程。 退回位置，带 类型＝1 用于单个轴（自软件版本SW6.4起） 退回位移，带 类型＝2 用于单个轴（自软件版本SW6.4起） 值也可以用类型＝0接收。仅仅是该值然后作为无效标记，并且用于退回必须重新编程。
POLF	指令POLF模态有效。
POLFA	如果一个轴不是单个轴，或者缺少类型或类型＝0，则会发出相应的报警26080和报警26081。

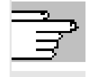
POLFMASK,	使用指令**POLFMASK**使能所说明的轴，用于退回，而没有轴之间的关联。指令**POLFMASK()**，没有说明一个轴，非激活快速退刀，用于所有退回时相互之间没有关联的轴。
POLFMLIN,	使用指令**POLFMLIN**使能所说明的轴，用于退回，轴之间具有线性关联。指令**POLFMLIN()**，没有说明一个轴，非激活快速退刀，用于所有退回时相互之间线性关联的轴。
achsnamei	轴名称，这些轴在LIFTFAST时应该运行到用POLF定义的位置。所有说明的轴必须处于同一个坐标系中。在通过**POLFMASK** 或者 **POLFMLIN** 可以使能快速退刀到一个固定位置之前，必须已经用POLF编程了一个位置，用于所选择的轴。没有给定机床数据用于预置**POLF**的值。 在插补**POLFMASK** 或者 **POLFMLIN**时，如果**POLF**还没有编程，则取消报警16016。

如果轴一个接一个地用POLFMASK, POLFMLIN 或者 POLFMLIN,POLFMASK使能，则对于该轴始终是最后的确定适用。



注意

*在零件程序开始处，用**POLF**编程的位置和通过 **POLFMASK** 或者**POLFMLIN**的激活被删除。*
*也就是说，用户必须在每个零件程序中重新编程用于 **POLF**和**POLFMASK** 或**POLFMLIN**所选择的轴的值。*



在功能描述M3中，您可以获得有关坐标系修改、取模回转轴的影响等等详细说明。



功能

停止

扩展停止的执行（NC控制）通过两个机床数据
MD 21380:ESR_DELAY_TIME1 和
MD 21381:ESR_DELAY_TIME2 说明。

轴不受干扰地继续插补在MD21380中的时间长度，
如同编程一样。在MD21380中时间段结束之后，引
入插补控制的制动（斜坡停止）。
对于插补控制的制动，在此可以使用MD21381中的时间
段，在此时间段之后引入带跟随运行的快速制动。
NC控制的停止

* 配置：通过 MD 37500:**21** 以及
  2 个时间参数，通过 MD s.o.;
* 使能 ($AA_ESR_ENABLE) 并且
* 启动：系统变量 $AC_ESR_TRIGGER
  用于单个轴，带 $AA_ESR_TRIGGER

举例说明单个轴的停止

`MD 37500: ESR_REACTION[AX1] = 22`	；NC控制的停止
`MD 21380: ESR_DELAY_TIME1[AX1] = 0.3`	
`MD 21381: ESR_DELAY_TIME2[AX1] = 0.06`	
`...`	
`$AA_ESR_ENABLE[AX1] = 1`	
`$AA_ESR_TRIGGER[AX1] = 1`	；从这里开始停止。

13.6.3 可能的触发源



功能

以下用于启动“扩展的停止和退回”的故障源可能：

* 公共源（NC外部/全局或者BAG/通道专用）：
  * 数字输入端（比如在NCU组件上或者在终端板上），或者数字输出端的、控制系统内部的、可读回的印象区
    ($A_IN, $A_OUT)
  * 通道状态($AC_STAT)
  * VDI-信号 ($A_DBB)
  * 一系列报警的总信号
    ($AC_ALARM_STAT)
* 轴向源：
  * 跟随轴的紧急退回阈（电子耦合同步运行，
    $VA_EG_SYNCDIFF[跟随轴]）



- 驱动：中间回路警告阈（危险的欠压），$AA_ESR_STAT[轴]
- 驱动：发生器－最低转速－阈（没有可回馈的旋转能量），$AA_ESR_STAT[轴]。

13.6.4 逻辑联系：源－反应－逻辑



功能

使用静态同步动作的柔性逻辑方法，从而可以根据源触发一定的反应。
用户可以决定利用静态同步动作逻辑联系所有相关的源。用户可以作为整体逻辑联系源－系统变量，或者利用位掩码也可以选择计算和联系所要求的反应。静态同步动作可以在所有运行方式生效。
如何使用同步动作的详细说明参见资料：/FBSY/
同步动作功能描述。

13.6.5 激活

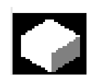


功能释放：
$AA_ESR_ENABLE
通过设置相关的控制信号 ($AA_ESR_ENABLE)
可以使能功能：发生器运行，停止，退回。该控制信号可以由同步动作修改。

功能触发（所有使能轴的共同释放）
$AN_ESR_TRIGGER

- 在识别出危险的中间回路欠压时，发生器运行“自动地”有效。
- 在识别出一个通讯故障时（在NC和驱动之间），以及在识别出运行时一个中间回路欠压时（前提条件配置和使能），激活驱动自给停止和/或退回。
- 另外，也可以从NC一侧，触发驱动自给的停止和/或退回，通过设置相应的控制信号 $AN_ESR_TRIGGER"（对所有驱动的广播指令）进行。

13.6.6 发生器运行/中间回路辅助



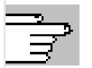
功能

通过对静态同步动作($AA_ESR_ENABLE)
设计驱动MD和相应的编程，可以短时间地补偿中间回路电压跌落。消除的时间取决于发生器存储的能量，它用于支持中间回路，并取决于保持当前运动所需要的能量（中间回路辅助和监控发生器转速极限）。



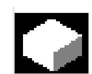


在低于中间回路电压下限时，相应轴/主轴由位置控制运行或转速控制运行转换到发生器运行。通过制动驱动（给定转速额定值＝0）能量反馈到中间回路。



详情参见
/FB/ M 3 , 联动, 引导值耦合

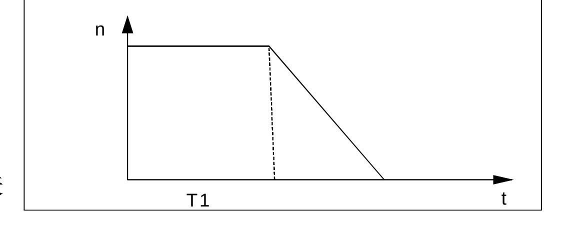
13.6.7 驱动自给停止



功能

一个事先耦合的关联驱动可以通过时间控制的关断延迟，以尽可能小的偏差相继停止，如果这在控制系统一侧不能实现。
驱动自给的停止通过MD进行配置和使能（在MD中的延迟时间T1），并且通过系统变量
$AA_ESR_ENABLE使能，通过
$AN_ESR_TRIGGER启动。

反应
在故障情况下正有效的转速给定值继续输出，时间T1。从而努力保持故障之前有效的运动，直至这种型式被取消，或者直至这种在其它驱动中发起的退回运动结束。这对于所有的引导驱动/跟随驱动，或者对于耦合中或关联中的驱动具有意义。



在时间T1之后，所有轴以转速给定值零－接通在电流极限处停止，并且在到达停止状态或在时间结束之后（＋驱动MD）删除脉冲。

13.6.8 驱动自给退回



功能

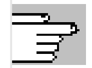
带数字611D驱动的轴可以（在设计时和使能时）

- 在系统故障时（生命符－识别）
- 在中间电压衰减低于一个报警限值时
- 在通过系统变量$AN_ESR_TRIGGER触发时

自己执行一个退回运动。

退回运动通过驱动611D自给进行。

自退回阶段开始时驱动就使其使能保持到先前有效的值。

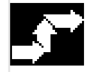


详情参见
/FB/ M 3 , 联动, 引导值耦合

13.6.9 举例：使用驱动自给的反应

配置举例

- 轴A应作为发生器驱动工作，
- 轴X在故障情况下以最大速度退回10毫米，并且
- 轴Y和Z延迟100毫秒停止，从而退回轴有时间去除机械耦合。



工作流程

1. 使能选件“扩展的停止和退回”和“运行方式搭接动作”（包含“静态同步动作IDS…”）。
2. 功能分配：
   $MA_ESR_REACTION[X]=11,
   $MA_ESR_REACTION[Y]=12,
   $MA_ESR_REACTION[Z]=12,
   $MA_ESR_REACTION[A]=10;
3. 驱动配置：

MD1639 RETRACT_SPEED[X]	=400000H	在正向（最大速度），
	=FFC00000H	在负向，
MD1638 RETRACT_TIME[X]	=10ms	(退回时间),
MD1637 GEN_STOP_DELAY[Y]	=100ms,	
MD1637 GEN_STOP_DELAY[Z]	=100ms,	
MD1635 GEN_AXIS_MIN_SPEED[A]	＝发生器－最低转速（转/分钟）。	

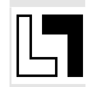
4. 功能释放（由零件程序或者同步动作）：
   $AA_ESR_ENABLE[X]=1,
   $AA_ESR_ENABLE[Y]=1,
   $AA_ESR_ENABLE[Z]=1,
   $AA_ESR_ENABLE[A]=1
5. 发生器驱动调到“摆动”转速
   （比如在主轴运行M03S1000）
6. 触发器条件作为静态同步动作表述，比如：
   - 取决于发生器轴的啮合：
     IDS=01 WHENEVER $AA_ESR_STAT[A]>0 DO
     $AN_ESR_TRIGGER=1
   - 和/或者取决于触发跟随运行的报警（位13＝2000H）：
     IDS=02 WHENEVER ($AC_ALARM_STAT B_AND
     'H2000')>0
     DO $AN_ESR_TRIGGER=1
   - 以及取决于EG同步运行监控（比如当Y作为EG跟随轴定义，最大允许的同步运行偏差100µm）：
     IDS=03 WHENEVER ABS($VA_EG_SYNCDIFF[Y])>0.1
     DO $AN_ESR_TRIGGER=1

13.7 链接－通讯 (自软件版本SW 5.2起)

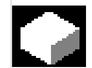


功能

在一台设备中各个NCU单元之间的NCU链接，
用于带分散式系统结构的设备中。在大量需要轴和通道
时，比如在回转台机床和多主轴机床中，计算效率，
配置方法以及带一个NCU的存储器区均已经到实际
极限。
几个与 NCU-链接-模块 链接的NCUs
提供一个敞开的方案，它们可以实现这种机床的任务。
该 NCU-链接-模块 (HW) 实现一个快速的 NCU-NCU-
通讯。



该功能与所订购的选件有联系。



功能

通过链接－模块相连的几个NCUs可以利用以下所描
述的系统变量－存取，对一个NCU全局存储器区进行
读写存取。

- 每个通过链接-模块相连的NCU
  可以统一使用可应答的 **全局**
  链接变量，用于所有相连的NCU。
- 链接-变量可以作为 系统变量编程。
  该变量的含义通常通过机床制造商确定和说明。
- 应用，用于链接－变量：
  - 全局机床状态
  - 工件装夹 敞开/闭合
  - 等等 ...
- 文件卷相对来说较小

- 传送速度很快，由此产生：这种使用为时间紧迫的信息设置。
- 对这些系统变量的存取可以从**零件程序** 和 **同步动作** 出发进行。用于NCU全局系统变量的存储器范围的大小可以设计。

在一个插补节拍
之后，所有参加的NCUs可以耐久地读出一个全局系统变量中新写入的值。

链接变量是**系统全局数据**，它们可以由所链接的NCUs作为 **系统变量**触发。这些
- 系统变量的内容，
- 它们的数据类型，
- 他们的使用,
- 它们在链接存储器中的位置（存取索引）
由用户（通常为机床制造商）确定。

链接变量存储在链接存储器中。
在引导之后链接存储器用0初始化。

在链接存储器之内可以触发下面的链接变量：
- INT $A_DLB[i] ；数据类型（8位）
- INT $A_DLW[i] ；数据字 (16 位)
- INT $A_DLD[i] ；数据双字(32 位)
- REAL $A_DLR[i] ；实际数据（64位）

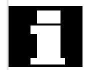
在读写链接变量时，根据相应类型触发1、2、4、8字节。

索引i表示相应变量的起始，与所设计的链接存储器的开始相关。索引自0开始计数。

值范围

下面的值范围与数据类型相联系：

BYTE:	0 到 255
WORD:	-32768 到 32767
DWORD:	-2147483648 到 2147483647
REAL:	-4.19e-308 到 4.19e-307

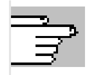


在某一个时刻，几个不同的NCUs共同对链接存储器存取的应用必须统一使用链接存储器。对于在时间上完全去除耦合的过程，该链接存储器可以不同地进行占用。



警告

只有当其它的NCUs也出现所写入的信息时，才结束一个链接变量的写过程。为此，需要大约两个插补节拍。为了保持稠度，局部地写入到链接存储器中延迟相同的时间。



其它的说明参见功能描述B3(SW5)。



编程举例

```
$A_DLB[5]=21
```

在公共链接存储器中第5个字节包含值21。

13.8 轴容器 (自软件版本 SW 5.2起)



功能

在回转台机床/多主轴机床中，夹装了工件的轴从一个加工单元运行到下一个加工单元。

因为加工单元位于不同的NCU通道中，在一个 站-/位置转换时，夹装了工件的轴必须重新动态地分配到相应的NCU通道中。 **轴容器**用于此目的。

在这个时刻始终只有一个工件夹装轴/主轴-在当地的加工单元中生效。轴容器组合了所有夹装轴/主轴的连接方法，这些当中始终只有一个激活，用于该加工单元。

通过轴容器可以分配：

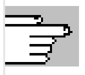
- 局部的轴 和/或
- 链接轴（参见基础部分）

通过轴容器定义的、可以使用的轴，在转换时通过推移到轴容器的登记来进行。

通过 **零件程序** 可以触发此推移。

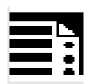
带链接轴的轴容器是一个NCU可以搭接的运行工具（NCU全局）,它通过控制系统进行协调。

只管理局部轴的轴容器是可以的。



有关轴容器的设计详细说明参见 /FB/，B3(SW5.2)。

在轴容器中登记时推移步距n可以用指令：



编程

AXCTSWE (CT_i)	AXIS CONTAINER SWITCH ENABLE
AXCTSWED(CT_i)	AXIS CONTAINER SWITCH ENABLE DIRECT



说明

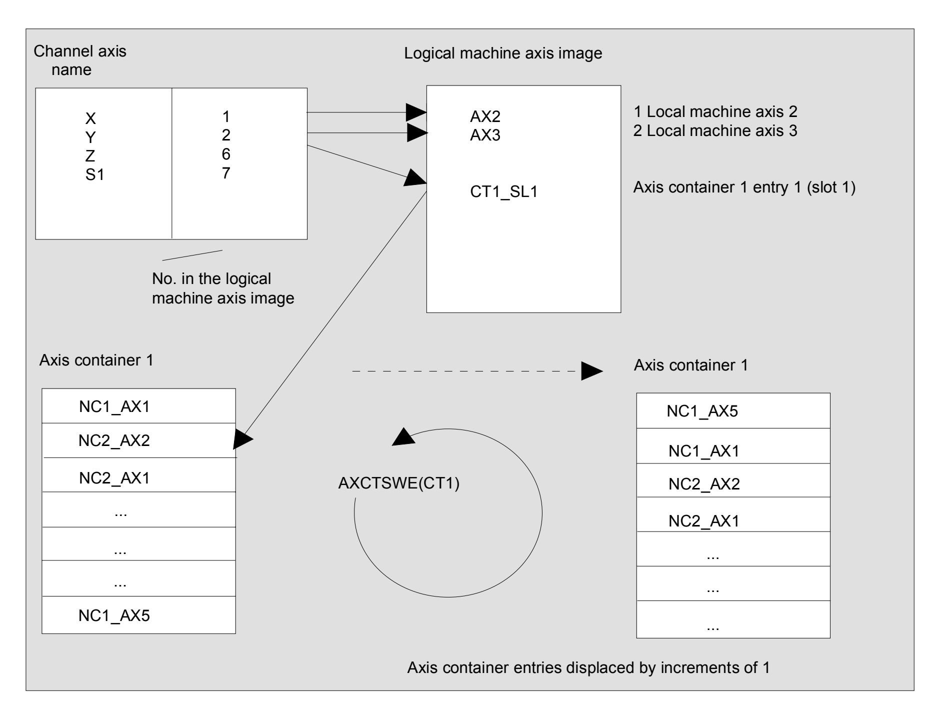
CT_i 或者	轴容器号，其内容应进行推移，或者
比如 A_CONT1	通过MD设置的轴容器的各个名称。



功能

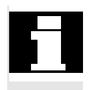
AXCTSWE ()

每个通道（通道轴在所说明的容器中登记）给出一个容器旋转的使能，当它结束位置/站加工时。如果在系统中用于容器轴的所有通道的使能均已经到达，则容器以在SD中存储的步距旋转。



在轴容器旋转1之后，在前面的举例中通道轴Z代替轴AX1分配到NCU1，轴AX5分配到NCU1。

指令变量AXCTSWED(CT_i)可以用于简化安装调试。轴容器在有效通道单独作用下旋转SD中所存储的步距。当剩余的通道（轴在容器中）处于**RESET**（复位）状态时，才允许使用该调用。



在轴容器旋转之后，所有的**NCUs**均有重新轴分配的问题，它们的通道通过逻辑加工轴图形指示到所旋转的轴容器。

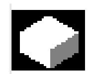
13.9 程序运行时间/工件计数器 (自软件版本 SW 5.2起)



功能

为了支持机床加工者，需要准备程序运行时间的信息和工件计数的信息。
这些信息在相应的机床数据中进行规定，并且可以作为NC或者PLC程序中的系统变量进行处理。同时这些信息供MMC在操作面板接口中使用。

13.9.1 程序运行时间



功能

在该功能中，定时器作为系统变量准备，它们可以用于监控工艺过程。
对于该定时器仅有读存取存在。MMC可以随时对它进行读存取。

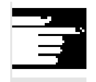


说明

下面两个定时器作为NCK专用的系统变量定义，并且一直有效。



$AN_SETUP_TIME	从最后的设置开始的时间，单位分钟。 在设置时复位
$AN_POWERON_TIME	从最后的PowerOn（上电）开始的时间，单位分钟 在PowerOn(上电）时复位
下面的三个定时器作为通道专用的系统变量定义，并且可以通过机床数据激活。	
$AC_OPERATING_TIME	在自动方式时NC程序的总的运行时间，单位秒
$AC_CYCLE_TIME	所选择的NC程序的运行时间，单位秒。
$AC_CUTTING_TIME	刀具啮合时间，单位秒。
$MC_RUNTIMER_MODE	刀具啮合时间，单位秒。



所有的定时器在系统引导时均用缺省值归零，并且可以与其激活无关而读出。

编程举例

1.激活运行时间测量，用于有效的NC程序，在有效的试运行进给和程序测试时没有进行测量：

```
$MC_PROCESSTIMER_MODE   =     'H2'
```

2.激活测量刀具啮合时间，在有效的试运行进给和程序测试时也进行测量：

```
$MC_PROCESSTIMER_MODE= 'H34'
```

3.激活测量总的运行时间和刀具啮合时间，在程序测试时也进行测量：

```
$MC_PROCESSTIMER_MODE= 'H25'
```

13.9.2 工件计数器



功能

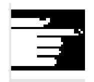
使用功能“工件计数器”可以准备计数器，比如可以用于系统内部的工件计数。这些计数器作为通道专用的系统变量存在，带读写存取，值范围为 0到999 999 999。

通过MDs可以对计数器激活、归零时间点和计数算术施加影响。

说明

准备以下的计数器：

$AC_REQUIRED_PARTS	所需工件的个数（工件给定值） 在此计数器中可以定义工件的个数，在到达这个数值之后，实际工件的个数$AC_ACTUAL_PARTS归零。通过一个MD可以激活显示报警“工件给定值到达”的产生，并且激活通道VDI信号“工件给定值到达”的产生。
$AC_TOTAL_PARTS	总共生产工件的个数（总实际值）该计数器说明自开始时刻后所生产的总的工件个数。该计数器仅在系统引导时自动地用缺省值归零。
$AC_ACTUAL_PARTS	当前工件的个数（当前实际值）在该计数器中登记了自开始时刻后所生产的所有工件的个数。在到达该工件给定值后 ($AC_REQUIRED_PARTS)，此计数器自动归零 (前提条件$AC_REQUIRED_PARTS 不等于 0 。)。
$AC_SPECIAL_PARTS	用户规定工件的个数。该计数器允许用户根据自己的定义进行工件计数。在与$AC_REQUIRED_PARTS (工件给定值) 一致时可以定义一个报警输出。计数器归零必须由用户自己进行。

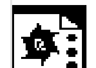


功能“工件计数器”与刀具管理功能无关。

所有的计数器可以由MMC读出并进行描述。

所有的计数器在系统引导时均用缺省值归零，并且可以与其激活无关而读写。





编程举例

1. 激活工件计数器 $AC_REQUIRED_PARTS:	
$MC_PART_COUNTER='H3'	$AC_REQUIRED_PARTS 有效，在 $AC_REQUIRED_PARTS == $AC_SPECIAL_PARTS 时显示报警
2. 激活工件计数器 $AC_TOTAL_PARTS:	
$MC_PART_COUNTER='H10' $MC_PART_COUNTER_MCODE[0]=80	$AC_TOTAL_PARTS 有效，使用M02 可以使计数器增加1，$MC_PART_COUNTER_MCODE[0] 没有含义
3. 激活工件计数器 $AC_ACTUAL_PARTS:	
$MC_PART_COUNTER='H300' $MC_PART_COUNTER_MCODE[1]=17	$AC_TOTAL_PARTS 有效，使用每个M17可以使计数器增加值1
4. 激活工件计数器 $AC_SPECIAL_PARTS:	
$MC_PART_COUNTER='H3000' $MC_PART_COUNTER_MCODE[2]=77	$AC_SPECIAL_PARTS 有效，使用每个M77可以使计数器增加值1
5. 关闭工件计数器 $AC_ACTUAL_PARTS:	
$MC_PART_COUNTER='H200' $MC_PART_COUNTER_MCODE[1]=50	$AC_TOTAL_PARTS 无效，Rest 没有意义
6.激活所有计数器，举例 1-4：	
$MC_PART_COUNTER ='H3313' $MC_PART_COUNTER_MCODE[0] =80 $MC_PART_COUNTER_MCODE[1] =17 $MC_PART_COUNTER_MCODE[2] =77	$AC_REQUIRED_PARTS 有效，在$AC_REQUIRED_PARTS == $AC_SPECIAL_PARTS时显示报警。 $AC_TOTAL_PARTS 有效，每次M02时计数器增加值1 $MC_PART_COUNTER_MCODE[0] 没有意义。 $AC_ACTUAL_PARTS 有效，每个M17时计数器增加值1 $AC_SPECIAL_PARTS 有效，每次M77时计数器增加值1

13.10 零件程序中交互式指令调用窗口：MMC (自软件版本 SW 4.4起)

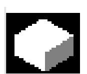


编程

```
MMC ("CYCLES, PICTURE_ON, T_SK.COM, BILD, MGUD.DEF, BILD_3.AWB, TEST_1,
A1","S")
```

说明

CYCLES	操作区，在此执行所设计的用户会话。
PICTURE_ON bzw. PICTURE_OFF	指令：屏幕选择或者屏幕撤销选择
T_SK.COM	Com文件：会话屏幕文件名称（用户循环）。在此确定会话屏幕的外观。在会话屏幕中可以显示用户变量和/或注释文本。
BILD	会话屏幕名称：单个的屏幕通过会话屏幕名称选择。
MGUD.DEF	用户数据定义文件，在读写变量时可以对此进行存取。
BILD_3.AWB	图形文件
TEST_1	显示时间或者应答变量
A1	文本变量...
"S"	应答方式：同步，通过软键“OK”进行应答

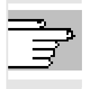


功能

通过指令MMC，用户定义的会话窗口（会话屏幕）可以从零件程序显示到MMC/HMI。

会话窗口的外观通过纯文本设计确定（在循环目录中的COM文件），MMC/HMI系统软件在此保持不变。

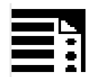
用户定义的会话窗口不可以同时在几个不同的通道中调用。



有关如何编程MMC指令的详细说明（包括编程示例），您可以在手册IM1到IM4中/IAM/部分找到，根据所使用的MMC/HMI软件而定。

13.11 运行控制的影响

13.11.1 按百分比的冲击补偿： JERKLIM



编程

```
JERKLIM[Achse]= ...
```



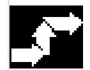
指令说明

JERKLIM	按照百分比修改所允许的最大冲击，它与机床数据中对该轴的设定值有关。
轴	加工轴，应对该轴的冲击极限值进行匹配。



功能

在程序各段处于临界状态时，有必要使冲击限制到最大可能的值之下，比如用于减小机床应力。加工方式SOFT必须有效。
该功能仅对轨迹轴有影响。



工作流程

在自动方式下，对于编程的轴，冲击极限值限制到机床数据中所存储冲击极限值的百分比值。

举例: N60 JERKLIM[X]=75
意义：轴溜板在X方向应以75%轴的允许冲击值进行加速/延迟。

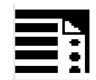
值范围： 1 ... 200
100表示：没有冲击影响。
100在复位和零件程序开始之后生效。



其它说明

在下一章节结束之后有另外一个示例。

13.11.2 成百分比的速度补偿： VELOLIM



编程

```
VELOLIM[Achse]= ...
```

指令说明

VELOLIM	按照百分比修改所允许的最大速度，它与机床数据中对该轴的设定值有关。
轴	加工轴，应对该轴的速度极限值进行匹配。



功能

在程序各段处于临界状态时，有必要使速度限制到最大可能的值之下，比如用于减小机床应力或者改善加工表面质量。该功能仅对轨迹轴和定位轴起作用。



工作流程

在自动方式下，对于编程的轴，速度极限值限制到机床数据中所存储速度极限值的百分比值。

举例: `N70 VELOLIM[X]=80`
意义：轴溜板在X方向应以80%轴的允许速度值进行运行。
值范围：1 ... 100
100表示：没有速度影响。
100在复位和零件程序开始之后生效。



编程举例

```
N1000 G0 X0 Y0 F10000 SOFT G64
N1100 G1 X20 RNDM=5 ACC[X]=20
      ACC[Y]=30
N1200 G1 Y20 VELOLIM[X]=5
      JERKLIM[Y]=200
N1300 G1 X0 JERKLIM[X]=2
N1400 G1 Y0
M30
```

13.12 主/从-联系

编程

`MASLDEF(Slv1, Slv2, ..., 主动轴)`	用于动态设计 （自软件版本SW6.4起）
`MASLDEL(Slv1, Slv2, ..., )`	用于动态设计 （自软件版本SW6.4起）
`MASLON(Slv1, Slv2, ..., )`	
`MASLOF(Slv1, Slv2, ..., )`	
`MASLOFS(Slv1, Slv2, ..., )`	（自软件版本SW6.4起）

参数说明

Slv1, Slv2, ...	从动轴，由一个主动轴控制
主动轴	控制有主/从联系定义的从动轴的轴



功能

在SW6.4之前的主/从耦合仅允许在参加轴静止时从动轴对主动轴的耦合。
SW6.4扩展版本允许与旋转的、转速控制的主轴耦合和分离，进行动态设计。

动态设计

MASLDEF
（自软件版本6.4起）
定义一个主/从连接，由零件程序出发。在SW6.4之前仅可以通过机床数据定义。

MASLDEL
（自软件版本SW 6.4起）
该指令取消从动轴到主动轴的分配，并且与MASLOF类似，同时分开当前的耦合。在机床数据中确定的主/从定义保持不变。

概述

MASLON
接通一个临时耦合

MASLOF
用此指令分开一个有效的耦合。
（自软件版本SW6.4起）
主轴在转速控制运行时，直接执行该指令。此时旋转的从动主轴维持其转速，直至重新编程转速。

MASLOFS
（自软件版本 SW 6.4起）
为了在分离耦合时可以进行自动制动，可以使用MASLOFS指令。当轴和主轴处于定位运行时，耦合必须在停止状态分离。



其它说明（自软件版本SW6.4起）

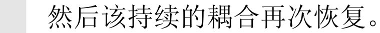
在MASLOF/MASLOFS时取消隐含的进刀停止。受缺少进刀停止的限制，用于从动轴的$P-系统变量不提供更新值，直至重新编程为止。



编程举例

动态设计一个主/从耦合，由零件程序出发：

在轴容器旋转以后，重要的轴应该成为主动轴。



MASLDEF(AUX,S3)	；S3主动，用于AUX
MASLON(AUX)	；耦合开，用于AUX
M3=3 S3=4000	；转动方向向右
MASLDEL(AUX)	；删除设计，并且分离耦合
AXCTSWE(CT1)	；容器旋转



其它说明（自软件版本SW6.4起）

从动轴可以通过PRESETON使实际值与主动轴的值相同而进行同步。为此，持续的主/从耦合必须短时间关断，从而使没有回参考点的从动轴的实际值通过POWER ON（上电）设置到主动轴的值。

然后该持续的耦合再次恢复。

该持续的主/从耦合用

MD 37262:MS_COUPLING_ALWAYS_ACTIVE = 1

激活，并且对临时耦合的语言指令没有影响。

编程举例

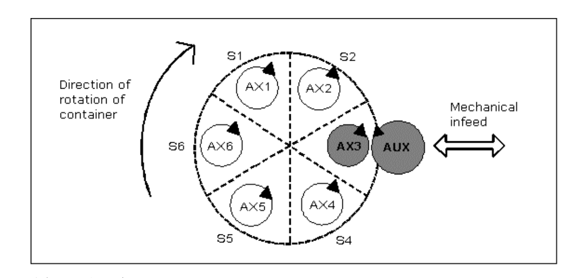
通过PRESETON使一个从动轴的实际值耦合到主动轴的相同值。

在一个持久的主/从耦合中，从动轴应该通过PRESETON修改其实际值。

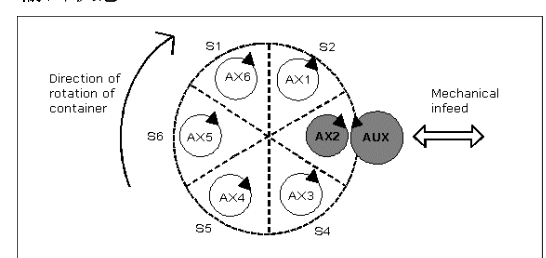
N37262 $MA_MS_COUPLING_ALWAYS_ACTIVE[AX2]=0	；持久的耦合短时间切断
N37263 NEWCONF	；
N37264 STOPRE	；
MASLOF(Y1)	；临时耦合关
N5 PRESETON(Y1, 0, Z1, 0, B1, 0, C1, 0, U1, 0)	；没有回参考点的从动轴设置实际值，因 ；为它们在PowerOn（上电）以后激活。
N37262 $MA_MS_COUPLING_ALWAYS_ACTIVE[AX2]=1	；激活持久的耦合
N37263 NEWCONF	

为了可以在容器旋转后用另一个主轴结束该耦合，必须事先分离旧的耦合，删除设计，并且设计一个新的耦合。

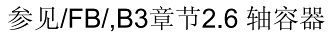
举例说明耦合顺序位置3/容器CT1



输出状态



在旋转一个槽之后


参见/FB/,B3章节2.6 轴容器

■

用于记录

独立的切削程序

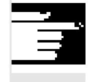
14.1 切削的辅助功能



自身的切削程序

您可以获得一个完整的加工循环用于切削。由此您可以用以下所叙述的功能编制自身的切削程序。

CONTPRON	打开以表格形式整理的轮廓（11栏）
CONTDCON	打开以表格形式译码的轮廓（6栏）
INTERSEC	计算两个轮廓单元的交点 。 （仅适用于用CONTPRON编制的表格）。
EXECTAB	程序段方式处理一个表格的轮廓单元 （仅适用于用CONTPRON编制的表格）。
CALCDAT	计算半径和圆心



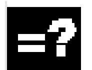
您不仅可以在切削时用这些功能，而且也可以用于其它场合。

14.2 轮廓预处理 - CONTPRON



编程

```
CONTPRON (TABNAME, BEARBART, NN, MODE)
EXECUTE (FEHLER)
```



参数说明

CONTPRON	打开轮廓预处理 。
TABNAME	轮廓表格的名称
BEARBART	加工方式参数： "G":纵向切削:内部加工 "L":纵向切削:外部加工 "N":车端面:内部加工 "P":车端面:外部加工
NN	咬边个数 ， 在整数型事件变量中
MODE （自软件版本SW4.4起）	加工方向, 类型INT 0 = 向前进行轮廓预处理 (如同目前的软件版本 SW 4.3, 缺省值） 1 = 在两个方向进行轮廓预处理
EXECUTE	结束轮廓预处理
FEHLER	报警应答变量, 类型INT 1 = 故障; 0 = 没有故障



功能

在CONTPRON之后运行的程序段描述预处理的轮廓。
这些程序段没有执行，而是存放在轮廓表格中。
每个轮廓单元相当于轮廓表格中二维数组的一个表格行。
所计算出的咬边个数送回。
用EXECUTE关断轮廓预处理，同时切换回通常的加工方式。
举例:

```
N30 CONTPRON(…)
N40 G1 X… Z…
N50…
N100 EXECUTE(…)
```



其它说明

调用的前提条件

在调用CONTPRON之前，必须

- 返回到一个可以无轮廓冲突进行加工的起始点
- 关断带G40的刀尖半径补偿。

允许的运行指令，坐标系

轮廓编程时仅允许使用指令G0到G3，附加倒圆和倒角。

自软件版本SW4.4起可以通过CIP和CT进行圆弧编程。

样条、多项式 螺纹功能引起出错。

不允许通过接通框架在CONTPRON 和EXECUTE之间改变坐标系。G70和G71/G700和G710之间的转换也同样适用。

在整理轮廓表格期间如果用GEOAX更换几何轴会导致报警。

预处理结束

通过调用EXECUTE（变量）可以在写轮廓之后切换回正常的程序运行，并且结束轮廓预处理。变量显示：

1 = 故障

0 = 没有故障（轮廓可以无错误地预处理）。

咬边单元

单个的咬边单元的轮廓描述既可以在一个子程序中进行，也可以在单个程序段中进行。

切削与所编程的轮廓方向无关（自软件版本SW4.4起）

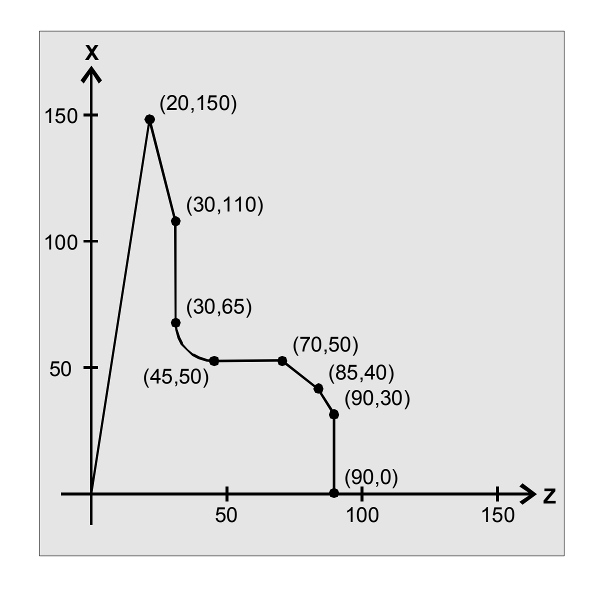
自软件版本SW4.4起，轮廓预处理CONTPRON可以进行扩展，从而使该轮廓表格在其调用之后可以使用，而与编程的方向无关。



编程举例 1

编制一个轮廓表格，带

- 名称 KTAB,
- 最多30个轮廓单元（圆弧，直线），
- 一个变量，表明所出现的咬边单元的个数
- 一个用于故障信息的变量



NC-零件程序

`N10 DEF REAL KTAB[30,11]`	轮廓表格，名称 KTAB ，比如最多30 个轮廓单元，参数11是一个固定的尺寸。
`N20 DEF INT ANZHINT`	变量，表示咬边单元的个数，名称 ANZHINT
`N30 DEF INT FEHLER`	用于应答的变量 0=没有故障, 1=故障
`N40 G18`	
`N50 CONTPRON (KTAB,"G",ANZHINT)`	调用轮廓预处理

`N60 G1 X150 Z20`
`N70 X110 Z30`
`N80 X50 RND=15`
`N90 Z70`
`N100 X40 Z85`
`N110 X30 Z90`
`N120 X0`
	N60 至 N120 轮廓描述

`N130 EXECUTE(FEHLER)`	结束轮廓表格的填充，转换到正常的程序运 行
`N140 …`	表格的继续加工





相关的表格KTAB

(0)	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)
7	7	11	0	0	20	150	0	82.40535663	0	0
0	2	11	20	150	30	110	-1111	104.0362435	0	0
1	3	11	30	110	30	65	0	90	0	0
2	4	13	30	65	45	50	0	180	45	65
3	5	11	45	50	70	50	0	0	0	0
4	6	11	70	50	85	40	0	146.3099325	0	0
5	7	11	85	40	90	30	0	116.5650512	0	0
6	0	11	90	30	90	0	0	90	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0

栏内容说明

(0) 指针到下一个轮廓单元（同一个行号）

(1) 指针到前一个轮廓单元

(2) 编码轮廓方式，用于运动

可能的值，用于X=abc

a = 10^2 G90 = 0 G91 = 1

b = 10^1 G70 = 0 G71 = 1

c = 10^0 G0 = 0 G1 = 1 G2 = 2 G3 = 3

(3), (4) 轮廓单元起始点

(3) = 横坐标, (4) = 纵坐标，在当前的平面中

(5), (6) 轮廓单元终点

(5) = 横坐标, (6) = 纵坐标，在当前的平面中

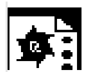
(7) 最大/最小指针:标记轮廓中局部的最大和最小

(8) 在轮廓单元和横坐标（纵向加工）或者纵坐标（车端面）之间的最大值 。

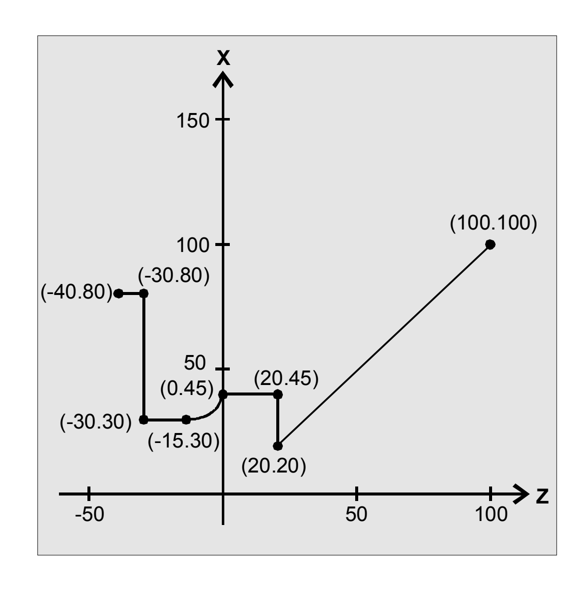
角度取决于所编程的加工方式

(9), (10) 如果是圆弧段，则轮廓单元的圆心坐标

(9) = 横坐标, (10) = 纵坐标



编程举例 2



编制一个轮廓表格，带

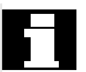
- 名称 KTAB,
- 最多92个轮廓单元（圆弧，直线），
- 工作方式纵向切削:外部加工
- 预处理向前和向后

NC-零件程序

`N10 DEF REAL KTAB[92,11]`	轮廓表格，名称 KTAB ，比如最多92个轮廓单元，参数11是一个固定的尺寸。
`N20 CHAR BT="L"`	工作方式，用于CONTPRON： 纵向切削，外部加工
`N30 DEF INT HE=0`	咬边单元=0的个数
`N40 DEF INT MODE=1`	预处理向前和向后
`N50 DEF INT ERR=0`	报警应答
`...`	
`N100 G18 X100 Z100 F1000`	
`N105 CONTPRON (KTAB, BT, HE, MODE)`	调用轮廓预处理

`N110 G1 G90 Z20 X20`
`N120 X45`
`N130 Z0`
`N140 G2 Z-15 X30 K=AC(-15) I=AC(45)`
`N150 G1 Z-30`
`N160 X80`
`N170 Z-40`	

`N180 EXECUTE(ERR)`	结束轮廓表格的填充，转换到正常的程序运行
`...`	



相关的表格KTAB

在结束轮廓预处理之后，可以在两个方向使用轮廓。

行	列（栏）										
	(0)	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)

0	6[1]	7[2]	11	100	100	20	20	0	45	0	0
1	0[3]	2	11	20	20	20	45	-3	90	0	0
2	1	3	11	20	45	0	45	0	0	0	0
3	2	4	12	0	45	-15	30	5	90	-15	45
4	3	5	11	-15	30	-30	30	0	0	0	0
5	4	7	11	-30	30	-30	45	-1111	90	0	0
6	7	0[4]	11	-30	80	-40	80	0	0	0	0
7	5	6	11	-30	45	-30	80	0	90	0	0
8	1[5]	2[6]	0	0	0	0	0	0	0	0	0
	...										
83	84	0[7]	11	20	45	20	80	0	90	0	0
84	90	83	11	20	20	20	45	-1111	90	0	0
85	0[8]	86	11	-40	80	-30	80	0	0	0	0
86	85	87	11	-30	80	-30	30	88	90	0	0
87	86	88	11	-30	30	-15	30	0	0	0	0
88	87	89	13	-15	30	0	45	-90	90	-15	45
89	88	90	11	0	45	20	45	0	0	0	0
90	89	84	11	20	45	20	20	84	90	0	0
91	83[9]	85[10]	11	20	20	100	100	0	45	0	0

栏内容说明

(0) 指针到下一个轮廓单元（同一个行号）

(1) 指针到前一个轮廓单元

(2) 编码轮廓方式，用于运动
可能的值，用于X=abc
a = 10^2 G90 = 0 G91 = 1
b = 10^1 G70 = 0 G71 = 1
c = 10^0 G0 = 0 G1 = 1 G2 = 2 G3 = 3

(3), (4) 轮廓单元起始点
(3) = 横坐标, (4) = 纵坐标，在当前的平面中

(5), (6) 轮廓单元终点
(5) = 横坐标, (6) = 纵坐标，在当前的平面中

(7) 最大/最小指针:标记轮廓中局部的最大和最小

(8) 在轮廓单元和横坐标（纵向加工）或者纵坐标（车端面）之间的最大值
角度取决于所编程的加工方式。

9), (10) 如果是圆弧段，则轮廓单元的圆心坐标。
(9) = 横坐标, (10) = 纵坐标

在栏中的注释说明

*始终在表格行**0**：*
1) 前一个：行n包含向前的轮廓结束
2) 后一个：行n是向前的轮廓表格结束

每一次在轮廓单元之内向前：
3) 前一个：轮廓起始（向前）
4) 后一个：轮廓结束（向前）

始终在轮廓表格行（向前）＋1：

5) 前一个：咬边向前个数
6) 后一个：咬边向后个数

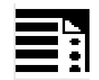
每一次在轮廓单元之内向后：

7) 后一个：轮廓结束（向后）
8) 前一个：轮廓开始（向后）

始终在最后的表格行：

9) 前一个：行n是轮廓表格起始（向后）
10) 后一个：行n包含轮廓起始（向后）

14.3 轮廓译码 - CONTDCON (自软件版本 SW 5.2起)



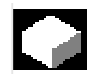
编程

```
CONTDCON (TABNAME,MODE)
EXECUTE (FEHLER)
```



参数说明

CONTDCON	打开轮廓预处理
TABNAME	轮廓表格的名称
MODE	加工方向，类型 INT 0 = 轮廓预处理 (缺省值)，根据轮廓段的顺序
EXECUTE	结束轮廓预处理
FEHLER	报警应答变量, 类型INT 1 = 故障; 0 = 没有故障



功能

在CONTDCON之后运行的程序段描述译码的轮廓。
这些程序段没有处理，而是译码后以占用存储器最小的方式存放在一个6栏的轮廓表格中。
每个轮廓单元相当于轮廓表格中的一个表格行。根据下面所说明的编码规则，可以由表格行DIN－编码－程序编排应用（比如循环）。在号码0的表格行中，存储输出点的数据。在待列表的程序块中，允许用于CONTDCON的G代码比功能CONTPRON中的范围更广。此外，每个轮廓段的进给和进给类型一起存储。
用EXECUTE关断轮廓预处理，同时切换回通常的加工方式。
举例:

```
N30 CONTDCON(…)
N40 G1 X… Z…
N50…
N100 EXECUTE(…)
```



其它说明

调用的前提条件

在调用CONTPRON之前，必须

- 返回到一个可以无轮廓冲突进行加工的起始点
- 关断带G40的刀尖半径补偿。

允许的运行指令，坐标系

下面的G组允许用于轮廓编程，带所说明的指令：

G-组 1:	G0, G1, G2, G3
G-组 10:	G9
G-组 11:	G60, G44, G641, G642
G-组 13:	G70, G71, G700, G710
G-组 14:	G90, G91
G-组 15:	G93, G94, G95, G96/G961

另外还有倒圆和倒角。

通过CIP和CT可以进行圆弧编程。样条、多项式螺纹功能引起出错。

不允许通过接通一个框架在CONTDCON 和EXECUTE之间改变坐标系。G70和G71/G700和G710之间的转换也同样适用。

在预处理轮廓表格期间如果用GEOAX更换几何轴会导致报警。

预处理结束

通过调用EXECUTE（故障），可以根据所写轮廓表格切换回正常的程序运行，并且结束轮廓预处理。在相应的变量FEHLER中进行应答：

0 = 没有故障(轮廓可以无错误地进行预处理)

1 = 故障

不允许的指令，错误的输出条件，重复的CONTDCON调用（不带EXECUTE()），太少的轮廓段或者表格定义得太小，所有这些均会额外导致报警。

在编程的轮廓方向切削

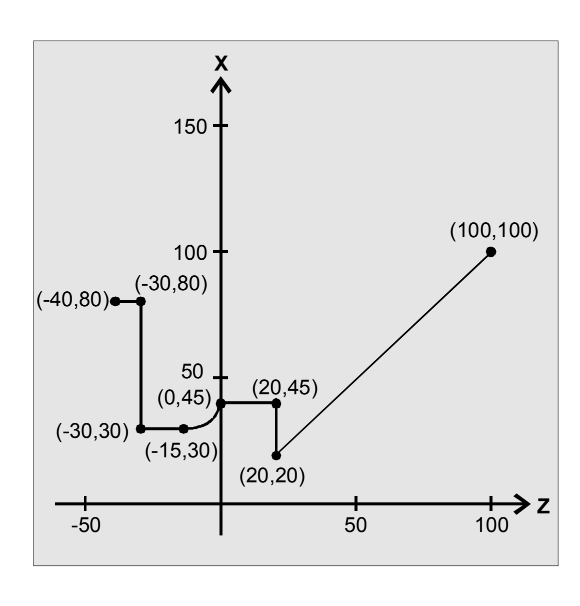
用CONTDCON产生的轮廓表格规定用于在轮廓编程方向进行切削。



编程举例

编制一个轮廓表格，带

- 名称 KTAB,
- 轮廓单元（圆弧，直线），
- 工作方式车削
- 预处理向前



NC-零件程序

`N10 DEF REAL KTAB[9,6]`	轮廓表格带名称 KTAB 和9个表格行。它允许8个轮廓程序段。 参数值6（表格的列数）是一个固定尺寸。
`N20 DEF INT MODE = 0`	缺省值0：仅在轮廓编程的方向。值1不允许。
`N30 DEF INT ERROR = 0`	报警应答
`...`	
`N100 G18 G64 G90 G94 G710`	
`N101 G1 Z100 X100 F1000`	
`N105 CONTDCON (KTAB, MODE)`	调用轮廓译码，MODE允许删除。

`N110 G1 Z20 X20 F200`
`N120 G9 X45 F300`
`N130 Z0 F400`
	轮廓描述

`N140 G2 Z-15 X30 K=AC(-15) I=AC(45)F100`	

`N150 G64 Z-30 F600`
`N160 X80 F700`
`N170 Z-40 F800`	

`N180 EXECUTE(ERROR)`	结束轮廓表格的填充，转换到正常的程序运行

...

列索引	0	1	2	3	4	5
行索引	轮廓方式	终点横坐标	终点纵坐标	圆心横坐标	圆心纵坐标	进给
0	30	100	100	0	0	7
1	11031	20	20	0	0	200
2	111031	20	45	0	0	300
3	11031	0	45	0	0	400
4	11032	-15	30	-15	45	100
5	11031	-30	30	0	0	600
6	11031	-30	80	0	0	700
7	11031	-40	80	0	0	800
8	0	0	0	0	0	0

列（栏）内容说明

行0　编码用于起始点：

列0：

10^0 (个位): G0 = 0

10^1 (十位): G70 = 0, G71 = 1, G700 = 2, G710 = 3

列1：　起始点横坐标

列2：　起始点纵坐标

列3－4：　0

列5　表格中最后轮廓段的行索引

行1-n：　登记轮廓段

列0：

10^0 (个位): G0 = 0, G1 = 1, G2 = 2, G3 = 3

10^1 (十位): G70 = 0, G71 = 1, G700 = 2, G710 = 3

10^2 (百位): G90 = 0, G91 = 1

10^3 (千位): G93 = 0, G94 = 1, G95 = 2, G96 = 3

10^4 (万位): G60 = 0, G44 = 1, G641 = 2, G642 = 3

10^5 (十万位): G9 = 1

列1：终点横坐标

列2：终点纵坐标

列3：圆心横坐标，在圆弧插补时

列4：圆心纵坐标，在圆弧插补时

列5：进给

14.4 两个轮廓单元的交点 - INTERSEC



编程

```
VARIB=INTERSEC (TABNAME1[n1], TABNAME2[n2], TABNAME3)
```



参数说明

VARIB	变量，用于状态　TRUE找到交点 FALSE:没有找到交点
TABNAME1[n1]	表格名称和n1。第一个表格的轮廓单元
TABNAME2[n2]	表格名称和n2。第二个表格的轮廓单元
TABNAME3	用于交点坐标的表格名称，在有效的平面G17-G19中



功能

INTERSEC从用 CONTPRON
产生的轮廓表格中计算两个统一的轮廓单元的交点。所说明的状态指出是否有一个交点（TRUE＝交点），或者没有交点（FALSE＝没有交点）。



其它说明

请注意：变量必须在其使用之前已经定义。



编程举例

计算表格KTAB1中的轮廓单元3与表格KTAB2中的轮廓单元7的交点。有效平面中的交点坐标存放在SCHNITT (第一个单元 = 横坐标, 第二个单元 = 纵坐标) 中。
如果没有交点，则跳跃到KEINSCH（没有找到交点）。

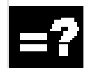
`DEF REAL KTAB1 [12, 11]`	轮廓表1
`DEF REAL KTAB2 [10, 11]`	轮廓表2
`DEF REAL SCHNITT [2]`	交点表
`DEF BOOL ISPOINT`	变量，用于状态
…	
`N10 ISPOINT=INTERSEC (KTAB1[3],KTAB2[7],SCHNITT)`	
	调用轮廓单元交点
`N20 IF ISPOINT==FALSE GOTOF KEINSCH`	跳跃到KEINSCH
…	

14.5 运行一个轮廓单元，从表格 EXECTAB中



编程

```
EXECTAB (TABNAME[n])
```



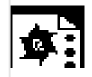
参数说明

TABNAME[n]	表格名称，带单元号n



功能

使用指令EXECTAB可以以程序段方式运行一个表格的轮廓单元，比如该单元由指令CONTPRON 产生。



编程举例

使用子程序 EXECTAB ，以程序段方式开始运行表格KTAB的轮廓单元。在调用时可以一个接一个把单元0到2交付使用。

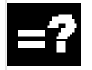
N10 EXECTAB (KTAB[0])	运行表格KTAB的单元0
N20 EXECTAB (KTAB[1])	运行表格KTAB的单元1
N30 EXECTAB (KTAB[2])	运行表格KTAB的单元2

14.6 计算圆弧数据 - CALCDAT



编程

```
VARIB = CALCDAT(PKT[n,2],ANZ,ERG)
```



参数说明

VARIB	变量，用于状态 TRUE = 圆弧, FALSE = 不是圆弧
PKT [n,2]	用于计算的点 n = 点数 (3 或者 4); 2 = 2个点坐标的说明
ANZ	用于计算的点的个数： 3 oder 4
ERG [3]	用于结果的变量：圆心坐标和半径的说明： 0 = 横坐标, 1 = 圆心纵坐标; 2 = 半径



功能

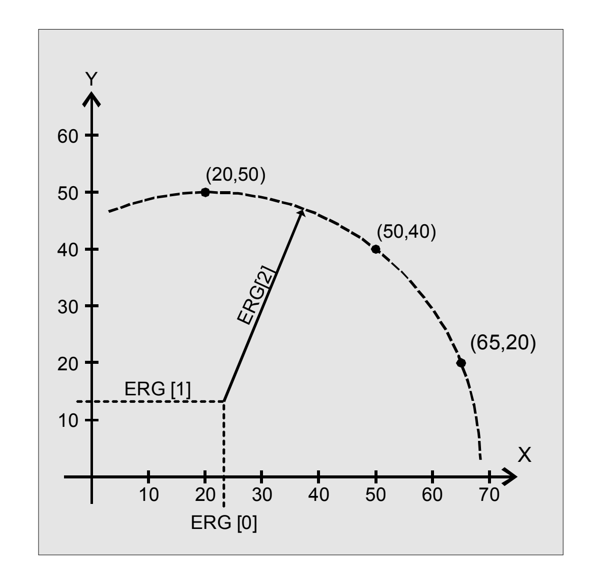
半径和圆心坐标由三个或者四个已知的圆弧点计算。
所给出的点必须不同。
如果是4个点，它们不是精确的在圆弧上，则选择一个平均值用于圆心和半径。





编程举例

由3个点计算出它们是否位于一个圆弧段。



N10 DEF REAL PKT[3,2]=(20,50,50,40,65,20)	点定义
N20 DEF REAL ERG[3]	结果
N30 DEF BOOL STATUS	变量，用于状态
N40 STATUS = CALCDAT(PKT,3,ERG)	调用求算的圆弧数据
N50 IF STATUS == FALSE GOTOF ERROR	跳转到故障

■

灵活的NC编程

15.1 指令表

图例说明：

[1] 程序开始时的基本设定（指系统供货时没有做其它的编程）。

[2] 组别编号与“编程基础部分”章节12.3中“G功能/准备功能”中的表一致。

[3] 绝对终点：模态方式；增量终点：程序段方式；否则模态方式/程序段方式取决于句法确定的G功能。

[4] IPO参数（增量）作为圆心。使用AC它们可以绝对编程。其它意义时（比如螺距）拒绝地址修改。

[5] 关键字不适用于 SINUMERIK FM-NC/810D。

[6] 关键字不适用于 SINUMERIK FM-NC/810D/NCU571。

[7] 关键字不适用于 SINUMERIK 810D。

[8] OEM用户可以另外再提出两个插补方式。.名称可以由OEM用户改变。

[9] 关键字仅适用于 SINUMERIK FM-NC。

[10] 对于该功能扩展的地址写入方式不允许。

名称	意义	赋值	说明，注释	句法	模态/程序段方式	组[2]
:	程序段号 – 主程序段 (参见 N)	0 ... 9999 9999 仅为整数，没有符号	特别的程序段标记，非N…；该程序段应包含所有用于后续加工段的指令	比如:20		
A	轴	实数			m,s [3]	
A2 [5]	刀具定向： 欧拉角	实数			s	
A3 [5]	刀具定向： 方向矢量	实数			s	
A4 [5]	刀具定向，用于程序段起始	实数			s	
A5 [5]	刀具定向，用于程序段结束：标准矢量	实数			s	
ABS	绝对值	实数				
AC	绝对尺寸输入	0, ..., 359.9999°		X=AC(100)	s	
ACC [5]	轴向加速度	实数，无符号			m	
ACN	回转轴绝对尺寸参数，在负方向返回位置			A=ACN(...) B=ACN(...) C=ACN(...)	s	
ACP	回转轴绝对尺寸参数，在正方向返回位置			A=ACP(...) B=ACP(...) C=ACP(...)	s	
ACOS	反余弦（三角函数）	实数				

ADIS	用于轨迹功能的精磨距离 G1, G2, G3, ...	实数，无符号			m	
ADISPOS	快速移动G0的精磨距离	实数，无符号			m	
ADISPOSA	用于IPOBRKA的公差窗口尺寸	整数，实数		ADISPOSA=..或者 ADISPOSA(<轴>[,REAL])	m	
ALF	快速退刀角度	整数，无符号			m	
AMIRROR	可编程镜像			AMIRROR X0 Y0 Z0 ；独立的程序段	s	3
AND	逻辑与					
ANG	轮廓线角度	实数				
AP	极角	0, ..., ± 360°			m,s [3]	
APR	读/显示存取保护	整数，无符号				
APW	写存取保护	整数，无符号				
AR	圆弧张角	0, ..., 360°			m,s [3]	
AROT	可编程旋转	旋转围绕第一几何轴 -180° ..180° 2. 几何轴： -89.999° .. 90° 3. 几何轴： -180° .. 180°		AROT X... Y... Z... ；独立 AROT RPL= 的程序段	s	3
AROTS	可编程的框架旋转，带立体角			AROT X... Y... AROT Z... X... AROT Y... Z... ；独立 AROT RPL= 的程序段	s	3
AS	宏指令定义	字符串				
ASCALE	可编程的比例			ASCALE X... Y... Z... ；独立的程序段	s	3
ASIN	反正弦（三角函数）	实数				
ASPLINE	Akima样条				m	1
ATAN2	反正切2	实数				
ATRANS	附加的、可编程的偏移			ATRANS X... Y... Z... ；独立的程序段	s	3
AX	变量轴名称	实数			m,s [3]	
AXCSWAP	容器轴转接			AXCSWAP(CTn,CTn+1,...)		25
AXIS	数据类型：进给轴名称		可以接收一个文件的名称			

AXNAME	把输入端字符串转换为轴名称	字符串	如果输入端字符串没有有效的轴名称，则设置一个报警			
AXSTRING	在SW5之前，转换为字符串的轴名称；自SW 6开始转换为字符串主轴号	在SW5之前AXIS，自SW6字符串	可以接收一个文件的名称	AXSTRING(SPI(n)) 自 SW 6 AXSTRING[SPI(n)]		
B	轴	实数			m,s [3]	
B_AND	位方式“与”					
B_NOT	位方式“非”					
B_OR	位方式“或”					
B_XOR	位方式“异－或”					
B2 [5]	刀具定向： 欧拉角	实数			s	
B3 [5]	刀具定向： 方向矢量	实数			s	
B4 [5]	刀具定向，用于程序段起始	实数			s	
B5 [5]	刀具定向，用于程序段结束；标准矢量	实数			s	
BAUTO	通过后面的3点确定第一个样条段				m	19
BLSYNC	在下一个程序段转换时才开始中断程序的加工。					
BNAT [1]	自然过渡到第一个样条程序段				m	19
BOOL	数据类型：真值TRUE/FALSE或者0/1					
BRISK [1]	突变型的轨迹加速度				m	21
BRISKA	接通突变型轨迹加速度，用于编程的轴					
BSPLINE	B 样条				m	1
BTAN	切线过渡到第一个样条程序段				m	19
C	轴	实数			m,s [3]	
C2 [5]	刀具定向：欧拉角	实数			s	
C3 [5]	刀具定向： 方向矢量	实数			s	
C4 [5]	刀具定向，用于程序段起始	实数			s	
C5 [5]	刀具定向，用于程序段结束；标准矢量	实数			s	
CAC	绝对坐标返回编码位置		编码的值是表索引；返回表数值			

CACN	在负方向绝对返回到表中存放的值 （编码位置绝对负方向）		允许用于编程回转轴作为定位轴			
CACP	在正方向绝对返回到表中存放的值。 （编码位置绝对正方向）					
CALCDAT	由3个或者4个点计算一个圆弧的半径和圆心（计算圆弧数据）	VAR Real [3]	必须区分这些点。			
CALL	间接子程序调用			CALL PROGVAR		
CALLPATH	子程序调用时编程的查找路径		可以编程一个带CALLPATH的路径，用于现在的NCK文件系统。	CALLPATH(/_N_WKS_DIR/_N_MYWPD/子程序名称_SPF)		
CANCEL	中止模态同步动作	INT	用所说明的ID中止。 没有参数：撤销选择所有模态的同步动作。			
CASE	有条件程序跳转					
CDC	直接返回编码位置		参见 CAC			
CDOF [1]	轮廓冲突检测“关”				m	23
CDON	轮廓冲突检测“开”				m	23
CDOF2	轮廓冲突检测“关”		仅用于 CUT3DC		m	23
CFC [1]	轮廓处恒定进给				m	16
CFIN	仅在内弯曲时恒定进给，在外弯曲时不恒定进给。				m	16
CFTCP	在刀尖基准点（圆心轨迹）恒定进给。				m	16
CHAN	规定数据有效区。		每个通道一次			
CHANDATA	设定通道号，用于通道数据存取	INT	仅允许在初始化模块中			
CHAR	数据类型：ASCII-字符	0, ..., 255				
CHECKSUM	通过一个字符串数组构成检查和，带一个确定的长度	Max. 长度32	提供16十六进制数字符串	ERROR=CHECKSUM		
CHF 自软件版本 SW 3.5 起 CHR	棱边；值＝棱边长度，在运动方向（倒棱） 棱边；值＝棱边长度	实数，无符号			S	
CHKDNO	D号的单一性检查					

CIC	增量坐标返回编码位置		参见 CAC			
CIP	通过中间点进行圆弧插补			CIP X... Y... Z... I1=... J1=... K1=...	m	1
CLEARM	复位一个/多个标号，用于通道协调	INT, 1 - n	不影响自身通道中的加工			
CLGOF	“关”恒定的工件转速，用于无心磨削					
CLGON	“开”恒定的工件转速，用于无心磨削					
CLRINT	撤销选择中断	INT	参数：中断号			
CMIRROR	对一个坐标轴的镜像	FRAME				
COARSEA	在到达“粗准停”时运动结束			COARSEA=..或者 COARSEA[n]=..	m	
COMPOF [1,6]	压缩器“关”				M...	30
COMPON [6]	压缩器“开”				m	30
COMPCURV	压缩器“开”：曲率恒定的多项式				m	30
COMPCAD	压缩器“开”：优化的表面品质				m	30
CONTPRON	轮廓准备“开”				m	49
COS	余弦（三角函数）	实数				
COUPDEF	定义ELG连接/同步主轴连接	字符串	程序段转换（SW）性能： NOC:没有SW控制， FINE / COARSE:SW在“精/粗同步运行” IPOSTOP:SW在给定值一侧结束叠加运动			
COUPDEL	删除ELG连接					
COUPOF	ELG连接/同步主轴副“关”					
COUPON	ELG连接/同步主轴副“开”					
COUPRES	复位ELG连接		编程的值无效：机床数据值有效			
CP	轨迹运行（连续路径）				m	49
CPRECOF [1,6]	可编程的轮廓精度“关”				m	39
CPRECON [6]	可编程的轮廓精度“开”				m	39
CPROT	通道专用的保护区“开/关”					

CPROTDEF	定义一个通道专用的保护区					
CR	圆弧半径	实数，无符号			S	
CROT	旋转当前坐标系	FRAME	Max. 参数个数： 6			
CROTS	可编程的框架旋转，带立体角			CROT X... Y... CROT Z... X... CROT Y... Z... ；独立的程序段 CROT RPL=	S	
CSCALE	比例系数，用于多个轴	FRAME	Max. 参数个数：2 * 轴数$_{max}$			
CSPLINE	立体样条				m	1
CT	圆弧，带切线过渡			CT X... Y.... Z...	m	1
CTAB	利用曲线表中引导轴位置，计算跟随轴位置	实数	参数4/5不可编程时：标准刻度			
CTABDEF	表格定义“开”					
CTABDEL	删除曲线表					
CTABEND	表格定义“关”					
CTABEXIST	检查带号n的曲线表		参数n			
CTABFNO	存储器中还可能的曲线表个数		存储器类型			
CTABFPOL	存储器中还可能的多项式个数		存储器类型			
CTABFSEG	存储器中还可能的曲线段个数		存储器类型			
CTABID	提供第n个曲线表的表格号		2个参数n和存储器类型			
CTABINV	利用曲线表中跟随轴位置，计算引导轴位置	实数	参见 CTAB			
CTABISLOCK	送回n号的曲线表的禁止状态		参数n			
CTABLOCK	设置禁止删除和改写		3 个参数 n, m, 和存储器类型			
CTABMEMTYP	送回存储器，在该存储器中编制了n号的曲线表		参数n			
CTABMPOL	存储器中最大可能的多项式个数		存储器类型			
CTABMSEG	存储器中最大可能的曲线段个数		存储器类型			
CTABNO	所定义的曲线表的个数，不管存储器类型		没有说明参数			
CTABNOMEM	在存储器SRAM或者DRAM中定义的曲线表的个数		存储器类型			
CTABPERIOD	送回n号的表格周期性		参数n			
CTABPOL	存储器中已经使用的多项式个数		存储器类型			
CTABPOLID	n号曲线表所使用的曲线多项式个数		参数n			

CTABSEG	已经使用的曲线段的个数	存储器类型			
CTABSEGID	n号曲线表所使用的曲线段个数	参数n			
CTABSEV	提供曲线表一个分段的跟随轴终值	通过LW确定段	R10 = CTABSEV(LW, n, 度, F轴, L轴)		
CTABSSV	提供曲线表一个分段的跟随轴起始值	通过LW确定段	R10 = CTABSSV(LW, n, 度, F轴, L轴)		
CTABTEP	在曲线表结束处提供引导轴的值	在曲线表结束处的引导值	R10 = CTABTEP(n, 度, L轴)		
CTABTEV	在曲线表结束处提供跟随轴的值	在曲线表结束处的跟随值	R10 = CTABTEV(n, 度, F轴)		
CTABTMAX	提供曲线表跟随轴的最大值	曲线表的跟随轴	R10 = CTABTMAX(n, F轴)		
CTABTMIN	提供曲线表跟随轴的最小值	曲线表的跟随轴	R10 = CTABTMIN(n, F轴)		
CTABTSP	在曲线表开始处提供引导轴的值	在曲线表开始处的引导值	R10 = CTABTSP(n, 度, L轴)		
CTABTSV	在曲线表开始处提供跟随轴的值	在曲线表开始处的跟随轴	R10 = CTABTSV(n, 度, F轴)		
CTABUNLOCK	取消禁止删除和改写	3 个参数 n, m, 和存储器类型			
CTRANS	零点偏移，用于多个轴 FRAME	最多8个轴			
CUT2D [1]	2 ½D 刀具补偿			m	22
CUT2DF	2 ½D 刀具补偿；刀具补偿相对于当前框架起作用（斜平面）。			m	22
CUT3DC [5]	3D 圆周铣削刀具补偿			m	22
CUT3DCC [5]	3D 圆周铣削刀具补偿，带限制面积			m	22
CUT3DCCD [5]	3D 圆周铣削刀具补偿，带差分刀具限制面积			m	22
CUT3DF [5]	3D 面铣削刀具补偿			m	22
CUT3DFF [5]	3D 面铣削刀具补偿，带恒定的刀具定向，与有效框架相关。			m	22
CUT3DFS [5]	3D 面铣削刀具补偿，带恒定的刀具定向，与有效框架无关。			m	22
CUTCONOF [1]	恒定半径补偿“关”			m	40
CUTCONON	恒定半径补偿“开”			m	40

D	刀具补偿号	1, ..., 9 自软件版本 SW 3.5 起 1, ... 32 000	包括确定刀具T…的补偿数据；D0 → 用于一个刀具的补偿值	D...		
DC	用于回转轴的绝对尺寸参数，直接返回位置			A=DC(...) B=DC(...) C=DC(...) SPOS=DC(...)	s	
DEF	变量定义	整数，无符号				
DEFAULT	跳转到CASE回路		如果表达式没有满足所说明的值，则跳转。			
DEFINE	宏指令定义					
DELAYFSTON	定义停止－延迟－范围的开始		隐含, 如果 G331/G332 有效		m	
DELAYFSTOF	定义停止－延迟－范围的结束				m	
DELDTG	删除剩余行程					
DELETE	删除所说明的文件。文件名可以用路径和文件标识给出。		可以删除所有文件			
DELT	删除刀具		可以删除Duplo号			
DIAMCYOF	G90/91的半径编程：开。显示时该组中最后有效的G代码有效。		半径编程最后有效的G代码		m	29
DIAMOF[1]	直径编程：关。		G90/G91半径编程		m	29
DIAMON	直径编程：开。		G90/G91的直径编程		m	29
DIAM90	G90的直径编程，G91的半径编程				m	29
DILF	快速退刀长度				m	
DISABLE	中断“关”					
DISC	过渡圆弧刀具半径补偿加强	0, ..., 100			m	
DISPLOF	抑制当前的程序段显示					
DISPR	Repos（再定位)-轨迹差值	实数，无符号			S	
DISR	Repos（再定位)-距离	实数，无符号			S	
DITE	螺纹导出位移	实数			m	
DITS	螺纹导入位移	实数			m	

DIV	整数－分割					
DL	刀具补偿和	INT			m	
DRFOF	关闭（取消）手轮偏移（DRF）				m	
DRIVE [9]	与速度相关的轨迹加速度				m	21
DRIVEA	开启折弯型加速度特征曲线，用于编程的轴					
DZERO	分配到通道的T0单元的所有刀具D号被设置无效。					
EAUTO	通过后面的3点确定最后一个样条段				m	20
EGDEF	定义一个电子齿轮		用于1个跟随轴，带最多5个引导轴。			
EGDEL	删除跟随轴的耦合定义。		触发进刀停止			
EGOFC	电子齿轮关断，连续					
EGOFS	电子齿轮关断，选择					
EGON	电子齿轮开通		无同步			
EGONSYN	电子齿轮开通		有同步			
EGONSYNE	电子齿轮开通，规定返回方式		有同步			
ELSE	当IF条件不满足时，程序跳转					
ENABLE	中断“开”					
ENAT [1,7]	曲线自然过渡到下一个运行程序段				m	20
ENDFOR	FOR-计数循环的结束行					
ENDIF	IF跳转的结束行					
ENDLOOP	程序死循环 LOOP 结束行					
ENDPROC	带起始行 PROC 的一个程序的结束行					
ENDWHILE	WHILE-循环的结束行					
ETAN	曲线切线过渡到下一个样条开始的运行程序段				m	20
EVERY	当条件从FALSE转换到TRUE时，执行同步动作					
EXECSTRING	给出一个字符串变量，带待执行的零件程序行		间接零件程序行	EXECSTRING(MFCT1 << M4711)		
EXECTAB	执行一个运行表中的单元					

EXECUTE	程序执行“开”		从基准预处理方式或者在构建一个保护区之后转换回到正常的程序处理			
EXP	指数函数e^x	实数				
EXTCALL	执行外部子程序		在“执行外部工作”方式下后装载HMI程序			
EXTERN	通告一个子程序，给出参数					
F	进给值（使用G4时，在F下也可以编程停留时间）	0.001, ..., 99 999.999	刀具/工件轨迹速度；单位毫米/分钟或者毫米/转取决于G94或者G95	F=100 G1 ...		
FA	轴向进给	0.001, ..., 999999.999 mm/min, Grad/min; 0.001, ..., 39999.9999 inch/min		FA[X]=100	m	
FAD	横向进给，用于返回运行和离开运行	实数，无符号				
FALSE	逻辑常量：错	BOOL	可以通过整数常量0代替			
FCTDEF	定义多项式函数		在 SYNFCT 或者 PUTFTOCF 中求值			
FCUB [6]	立体样条进给，可改变		作用于带G93和G94的进给		m	37
FD	轨迹进给，用于手轮叠加	实数，无符号			s	
FDA	轴向进给，用于手轮叠加	实数，无符号			s	
FENDNORM	拐角延迟“关”				m	57
FFWOF [1]	预控制“关”				m	24
FFWON	预控制“开”				m	24
FIFOCTRL	进刀运行缓冲控制				m	4
FIFOLEN	可编程的进刀深度					

FINEA	在到达“精准停”时运动结束			FINEA=...或者 FINEA[n]=..	m	
FL	同步轴极限速度	实数，无符号	用G93,G94,G95设定的单位有效（最大快速移动）	FL [轴] =...	m	
FLIN [6]	进给线性改变		作用于带G93和G94的进给		m	37
FMA	多次轴向进给	实数，无符号			m	
FNORM [1,6]	正常进给，符合DIN66025				m	37
FOCOF	关闭带极限力矩/力的运行				m	
FOCON	开通带极限力矩/力的运行				m	
FOR	带固定运行次数的计数循环					
FORI1	以大圆进给，用于方向矢量转向				m	
FORI2	进给，用于围绕转向的方向矢量的叠加旋转				m	
FP	固定点：待返回运行固定点的号	整数，无符号		G75 FP=1	S	
FPO	通过多项式编程的进给过程	实数	平方、立方的多项式系数			
FPR	回转轴标记	0.001 ... 999999.999		FPR (回转轴)		
FRAME	数据类型，用于确定坐标系		每个几何轴包含：进给，旋转，位移角，刻度，镜像； 每个附加轴包含：进给，刻度，镜像			

FRC,	进给，用于半径和棱边				s	
FRCM,	进给，用于半径和棱边模态				m	
FTOC	修改刀具精补偿		取决于一个用FCTDEF确定的3次多项式			
FTOCOF[1,6]	在线有效的刀具精补偿“关”				m	33
FTOCON[6]	在线有效的刀具精补偿“开”				m	33
FXS	运行到固定挡块	整数，无符号	1 = 选择, 0 = 撤销选择		m	
FXST	运行碰到固定挡块的力矩极限	%	可选参数		m	
FXSW	运行碰到固定挡块的监控窗口	毫米，英寸或者度	可选参数			

G功能						
G	G功能（准备功能） G功能分为各个G组。在一个程序段中，仅能写入一个G组中的一个G功能。 一个G功能可以模态有效（直至通过同一组中的另一个G功能撤换），或者它仅对所在程序段有效（程序段方式有效）。	仅为整数，规定值		G...		
G0	线性插补，带快速移动		运行	G0 X... Z...	m	1
G1[1]	线性插补，带进给		指令	G1 X... Z... F...	m	1
G2	顺时针圆弧插补			G2 X... Z... I... K... F... ；圆心和终点 G2 X... Z... CR=... F... ；圆心和终点 G2 AR=... I... K... F... ；张角 和圆心 G2 AR=... X... Z... F... ；张角 和终点	m	1
G3	逆时针圆弧插补			G3 ... ；其它同G2时	m	1
G4	停留时间，给定时间		特殊运行	G4 F...；停留时间， 以S；或者 G4 S...；停留时间，以 主轴转数 ；独立的程序段	s	2
G9	准停速度取消				s	11
G17[1]	选择工作平面X/Y		横向进给方向Z		m	6

G18	选择工作平面Z/X		横向进给方向Y		m	6
G19	选择工作平面Y/Z		横向进给方向X		m	6
G25	工作区域限制下限		赋值，	G25 X.. Y..Z..;独立的程序段	s	3
G26	工作区域限制上限		在通道轴中	G26 X.. Y..Z..;独立的程序段	s	3
G33	螺纹插补，带恒定螺距	0.001, ..., 2000.00 mm/U	运行指令	G33 Z... K... SF=... ；圆柱螺纹 G33 X... I... SF=... ；平面螺纹 G33 Z... X... K... SF=... ；圆锥螺纹（Z轴位移大于X轴位移平面螺纹） G33 Z... X... I... SF=... ；圆锥螺纹（X轴位移大于Z轴位移）。	M...	1
G34	螺距增加（递增修改）		运行指令	G34 Z... K... F_{ZU}=...	m	1
G35	螺距减小（递减修改）		运行指令	G35 Z... K... F_{AB}=...	m	1
G40 [1]	刀具半径补偿“关”				m	7
G41	刀具半径补偿，轮廓左边				m	7
G42	刀具半径补偿，轮廓右边				m	7
G53	抑制当前零点偏移（程序段方式）		包括编程的偏移		s	9
G54	1. 可设定的零点偏移				m	8
G55	2. 可设定的零点偏移				m	8
G56	3. 可设定的零点偏移				m	8
G57	4. 可设定的零点偏移				m	8
G58	可编程的偏移		轴向替换		s	3
G59	可编程的偏移		附加轴向替换		s	3
G60 [1]	准停速度取消				m	10
G62	内角处拐角延迟，在刀具半径补偿有效时（G41,G42)		仅与轨迹控制运行一起	G62 Z... G1	m	57
G63	攻丝，带补偿夹具			G63 Z... G1	s	2
G64	准停－轨迹控制运行				m	10
G70	英制尺寸（长度）				m	13
G71 [1]	公制尺寸（长度）				m	13
G74	回参考点运行		加工轴	G74 X...Z..; 独立的程序段	s	2
G75	回固定点运行			G75 FP=.. X1=...；独立的程序段	s	2
G90 [1]	绝对尺寸输入			G90 X... Y... Z...(...)Y=AC(...) 或者 X=AC Z=AC(...)	m s	14
G91	增量尺寸输入			G91 X... Y... Z...或者 X=IC(...) Y=IC(...) Z=IC(...)	m s	14

G93	时间倒数进给1/分钟		一个程序段的开始运行：时间长度	G93 G01 X... F...	m	15
G94[1]	线性进给F，单位毫米/分钟或者英寸/分钟和度/分钟				m	15
G95	转速进给F，单位毫米/转或者英寸/转				m	15
G96	恒定切削速度“开”			G96 S... LIMS=... F...	m	15
G97	恒定切削速度“关”				m	15
G110	极点编程，相对于最后编程的给定位置			G110 X.. Y.. Z..	s	3
G111	极点编程，相对于当前工件坐标系的零点			G110 X.. Y.. Z..	s	3
G112	极点编程，相对于最后有效的极点			G110 X.. Y.. Z..	s	3
G140[1]	确定返回运行方向WAB，通过G41/G42				m	43
G141	返回运行方向WAB，轮廓左边				m	43
G142	返回运行方向WAB，轮廓右边				m	43
G143	返回运行方向WAB，切线相关				m	43
G147	转换以直线返回运行				s	2
G148	转换以直线开始运行				s	2
G153	抑制当前框架，包括基准框架				s	9
G247	转换以1/4个圆弧返回运行				s	2
G248	转换以1/4个圆弧开始运行				s	2
G331	攻丝	± 0.001, ...,	运行		m	1
G332	退回（攻丝）	2000.00 mm/U	指令		m	1
G340[1]	空间返回运行程序段 （深度和平面中同时）（螺旋）		在转换返回运行/开始运行时生效		m	44
G341	首先在垂直轴上横向进给(z)，然后在平面中返回运行		在转换返回运行/开始运行时生效		m	44
G347	转换以半圆返回运行				s	2
G348	转换以半圆开始运行				s	2
G450[1]	过渡圆弧		拐角特性，		M...	18
G451	等距离交点		在刀具半径补偿时		m	18
G460[1]	在WRK时返回运行/开始运行特性				m	48
G461	在WRK时返回运行/开始运行特性				m	48
G462	在WRK时返回运行/开始运行特性				m	48
G500[1]	如果在G500中没有值，则关断所有可设定框架。				m	8
G505 .... G599	5. ... 99. 可设定的零点偏移				m	8
G601[1]	在精准停时程序段转换		有效，仅当		m	12
G602	在粗准停时程序段转换		G60有效时，或者		m	12
G603	在IPO程序段结束处程序段转换		G9带可编程的		m	12
G641	准停－轨迹控制运行		过渡磨削	G641 ADIS=...	m	10
G642	精磨削，带轴向精度				m	10

G643	程序段内部精磨削				m	10
G644	精磨削，给定轴动态				m	10
G621	在所有角处拐角延迟		仅与轨迹控制运行一起	G621 ADIS=...	m	57
G700	尺寸参数，英寸和英寸/分钟（长度＋速度＋系统变量）				m	13
G710[1]	公制尺寸参数，毫米和毫米/分钟（长度＋速度＋系统变量）				m	13
G810[1], ..., G819	给OEM用户保留的G代码组					31
G820[1], ..., G829	给OEM用户保留的G代码组					32
G931	进给规定，通过运行时间		运行时间		m	15
G942	冻结线性进给和恒定切削速度或者主轴转速				m	15
G952	冻结转速进给和恒定切削速度或者主轴转速				m	15
G961	恒定切削速度“开”		进给类型同G94时	G961 S... LIMS=... F...	m	15
G962	线性进给或者转速进给和恒定切削速度				m	15
G971	恒定切削速度“关”				m	15
G972	冻结线性进给或者转速进给和恒定主轴转速				m	15
GEOAX	给几何轴1－3分配新的通道轴		没有参数：MD确定生效			
GET	占用加工轴（n)		轴必须用RELEASE在其它通道中使能。			
GETD	直接占用加工轴（n)		参见GET			
GETACTT	从相同名称的刀具组中确定一个有效的刀具					
GETSELT	提供一个预选的T号					
GETT	给刀具名确定T号					
GOTOF	跳转指令，向前（程序结束方向）					
GOTOB	跳转指令，向后（程序开始方向）					
GOTO	跳转指令，先向前后向后（先向程序结束方向，然后向程序开始方向）					
GOTOC	抑制报警14080 跳转目标没有找到		参见GOTO			
GWPSOF	撤销选择恒定砂轮圆周速度(SUG)			GWPSOF (T-号)	s	
GWPSON	选择恒定砂轮圆周速度(SUG)			GWPSON (T-号)	s	
H...	辅助功能，到PLC	实数/整数	通过MD可以设定（机床制造商）	H100 或者 H2=100		

I [4]	插补参数	实数			s	
I1	中间点坐标	实数			s	
IC	增量尺寸输入	0, ..., ±99999.999°		X=IC(10)	s	
IDS	静态同步动作标记					
IF	引入一个有条件跳转		结构： IF - ELSE - ENDIF			
INDEX	确定输入端字符中一个符号的索引	0, ..., INT	字符串：1. 参数，符号：2. 参数			
INIT	选择模块，用于在一个通道中的加工					
INT	数据类型：带符号的整数值	$-(2^{31}-1)$, ..., $2^{31}-1$				
INTERSEC	计算两个轮廓单元之间的交点	VAR REAL [2]	故障状态： BOOL			
IP	插补参数变量	实数				
IPOBRKA	运行准则，自制动斜坡开始点		制动斜坡，在100%到0%时	IPOBRKA=..或者 IPOBRKA(<轴>[,实数])	m	
IPOENDA	在到达“IPO停止”时运动结束			IPOENDA=..或者 IPOENDA[n]..	m	
IPTRLOCK	冻结不可查找的程序段到下一个加工功能程序段的开始。		冻结中断指针		m	
IPTRUNLOCK	设置不可查找的程序段到当前的程序段的结束，用于中断时刻		设置中断指针		m	
ISAXIS	检查是否有作为参数尺寸的几何轴1－3	BOOL				
ISD	插入深度	实数			m	
ISFILE	检查在NCK用户存储器中是否有一个文件。	BOOL	提供一个布尔型结果	RESULT=ISFILE("Testfile“) IF (RESULT==FALSE)		
ISNUMBER	检查是否可以把输入端字符串转换成数字	BOOL	转换输入端字符串为数字			
ISVAR	检查传送参数是否包含一个NC知晓的变量	BOOL	机床数据, 设定数据和变量, 同GUD‘s			
J [4]	插补参数	实数			s	
J1	中间点坐标	实数			s	
JERKA	激活通过MD设定的加速度特性，用于编程的轴					
K [4]	插补参数	实数			s	
K1	中间点坐标	实数			s	
KONT	在刀具补偿时绕行轮廓				m	17
KONTC	用曲率不变的多项式返回运行/开始运行				m	17
KONTT	用切线不变的多项式返回运行/开始运行				m	17
L	子程序号	整数，最大7个数		L10	s	
LEAD [5]	超前角	实数			m	

LEADOF	引导值耦合“关”					
LEADON	引导值耦合“开”					
LFOF [1]	螺纹切削中断“关”				m	41
LFON	螺纹切削中断“开”				m	41
LFTXT [1]	切向退刀时的刀具方向				m	46
LFWP	非切向退刀时的刀具方向				m	46
LFPOS	轴向退刀到一个位置				m	46
LIFTFAST	在调用中断程序之前快速退刀					
LIMS	在G96/G961和G97时转速极限（主轴速度极限）	0.001 ... 99 999.999			m	
LN	自然对数	实数				
LOCK	禁止带ID的同步动作（停止工艺循环）					
LOG	（十位一）对数	实数				
LOOP	引入一个无限循环		结构： LOOP - ENDLOOP			
M...	开关操作	0, ..., 9999 9999	max. 由机床制造商确定最大5个自由的M功能			
M0 [10]	编程停止					
M1 [10]	可选停止					
M2 [10]	主程序程序结束，复位到程序开始					
M3	主轴右旋方向，用于主主轴					
M4	主轴左旋方向，用于主主轴					
M5	主轴停止，用于主主轴					
M6	换刀					
M17 [10]	子程序结束					
M19	主轴定位					
M30 [10]	程序结束，同M2					
M40	自动换档					
M41... M45	齿轮级1,…,5					
M70	过渡到轴运行					
MASLDEF	定义主/从轴连接					
MASLDEL	分离主/从轴连接，删除连接定义					
MASLOF	关闭一个临时耦合					
MASLOFS	使用自动制动从动轴，关闭一个临时耦合					
MASLON	接通一个临时耦合					
MCALL	模态子程序调用		没有 子程序名:撤销选择			

MEAC	连续测量，没有删除剩余行程	整数， 无符号			S	
MEAFRAME	由测量点计算框架	FRAME				
MEAS	用卡规测量	整数， 无符号			S	
MEASA	测量，带剩余行程删除				s	
MEAW	用卡规测量，没有剩余行程删除	整数，无符号			S	
MEAWA	测量，不删除剩余行程				s	
MI	存取框架数据：镜像					
MINDEX	确定输入端字符串中一个符号的索引	0, ..., INT	字符串：1. 参数， 符号：2. 参数			
MIRROR	可编程的镜像			MIRROR X0 Y0 Z0 ；独立的程序段	s	3
MMC	从零件程序中交互式调用对话窗口 MMC/HMI	字符串				
MOD	取模除法					
MOV	启动定位轴	实数				
MSG	可编程的信息			MSG("信息")	m	
N	程序段号－副程序段	0, ...,9999 9999 仅为整数，没有符号	可以用于标记带号的程序段；位于一个程序段的起始处	比如N20		

NCK	规定数据有效区。		每个通道一次			
NEWCONF	接收修改的机床数据。根据机床数据设置有效。		也可以通过HMI软键			
NEWT	编制新的刀具		可以删除Duplo号			
NORM [1]	在刀具补偿时，在起始点和终点处正常设定				m	17
NOT	逻辑“非”					
NPROT	机床专用的保护区“开/关”					
NPROTDEF	定义一个机床专用的保护区					
NUMBER	转换输入端字符串为数字		实数			
OEMIPO1[6,8]	OEM插补1				m	1
OEMIPO2[6,8]	OEM插补2				m	1
OF	CASE回路中的关键字					
OFFN	编程轮廓的加工余量			OFFN=5		
OMA1 [6]	OEM地址1	实数			m	

OMA2[6]	OEM地址2	实数			m	
OMA3[6]	OEM地址3	实数			m	
OMA4[6]	OEM地址4	实数			m	
OMA5[6]	OEM地址5	实数			m	
OFFN	偏移补偿－正常	实数			m	
OR	逻辑“或”					
ORIC[1,6]	在外角的方向改变叠加到待插入的圆弧程序段。				m	27
ORID[6]	方向改变在圆弧程序段之前执行				m	27
ORIAXPOS	虚拟的方向轴与回转轴位置的方向角				m	50
ORIEULER	欧拉角方向角				m	50
ORIAXES	线性插补加工轴或者方向轴		终点方向： 矢量参数 A3, B2, C2 附加参数： 旋转矢量 A6, B6, C6 锥形张角， 单位度： 0<NUT<180 度 中间矢量： A7, B7, C7 2. 刀具触点： XH, YH, ZH	参数如下： 方向矢量 标准化 A6=0, B6=0, C6=0 张角作为运行角，带 NUT=... NUT=+...在 ≤ 180 度时 NUT= -... 在 ≥ 180 度时 中间方向 标准化 A7=0, B7=0, C7=1	m	51
ORICONCW	顺时针方向圆弧表面插补				m	51
ORICONCCW	逆时针方向圆弧表面插补				m	51
ORICONIO	圆弧表面插补，说明一个中间方向				m	51
ORICONTO	圆弧表面插补，在切线过渡时（终点方向说明）				m	51
ORICURVE	方向插补，规定刀具两个触点的运动				m	51
ORIPLANE	平面中插补（根据ORIVECT）				m	51
ORIPATH	刀具定向路径与轨迹有关		处理转换软件包，参见/FB/,TE4		m	51
ORIRPY	欧拉角方向角				m	50
ORIS[5]	方向改变	实数	与轨迹有关		m	
ORIVECT	大圆插补（与ORIPLANE一致）				m	51
ORIVIRT1	方向角，通过虚拟的方向轴（定义1）				m	50
ORIVIRT2	方向角，通过虚拟的方向轴（定义1）				m	50
ORIMKS[6]	在机床坐标系中刀具定向				m	25
ORIWKS[1,6]	在工件坐标系中刀具定向				m	25
OS	摆动“开/关”	整数， 无符号				
OSC[6]	恒定平滑修改刀具方向				m	34
OSCILL	轴配置用于摆动 － 开通摆动		轴：1－3 横向进给轴		m	

OSCTRL	选件摆动	整数，无符号			M	
OSE	摆动：终点				m	
OSNSC	摆动：火花放电次数				m	
OSOF [1,6]	平滑修整刀具方向“关”				m	34
OSP1	摆动：左换向点	实数			m	
OSP2	摆动：右换向点	实数			m	
OSS [6]	在程序段结束处平滑刀具方向				m	34
OSSE [6]	在程序段开始处和结束处平滑刀具方向				m	34
OST1	摆动：停止点，在左换向点	实数			m	
OST2	摆动：停止点，在右换向点	实数			m	
OVR	转速补偿（倍率）	1, ..., 200%			m	
OVRA	轴向转速补偿（倍率）	1, ..., 200%			m	

P	零件程序运行次数	1 ... 9999, 整数，无符号		比如 L781 P... ；独立的程序段		
PCALL	用绝对的路径参数和参数传送调用子程序		没有绝对的路径特性，如同CALL			
PDELAYOF [6]	冲压时延迟“关”				m	36
PDELAYON [1,6]	冲压时延迟“开”				m	36
PL	参数－间隔－长度	实数，无符号			S	
PM	每分钟			每分钟进给		
PO	多项式	实数，无符号			S	
POLF	位置 LIFTFAST	实数，无符号	在WKS中的几何轴，其它MKS	POLF[Y]=10 退回轴的目标位置	m	
POLFA	用 $AA_ESR_TRIGGER 启动单个轴的退回位置		用于单个轴	POLFA(AX1, 1, 20.0)	m	
POLFMASK	轴使能，用于退回，轴之间没有相互关系		已选择的轴	POLFMASK(AX1, AX2, ...)	m	
POLFMIN	轴使能，用于退回，轴之间有线性关系		已选择的轴	POLFMIN(AX1, AX2, ...)	m	
POLY [5]	多项式插补				m	1
POLYPATH [5]	多项式插补可选择，用于轴组AXIS或者VECT			POLYPATH ("AXES") POLYPATH ("VECT")	m	1
PON [6]	冲压“开”				m	35
PONS [6]	冲压“开”，以IPO节拍				m	35
POS	轴定位			POS[X]=20		

POSA	轴定位，超出程序段界限			POSA[Y]=20		
POSP	分段定位（摆动）	实数：终点位置，分段长度； 整数：选件				
POT	平方（算术函数）	实数				
PR	每转			转速进给		
PRESETON	实际值设定，用于编程的轴		每次编程一个轴名称，并在下一个参数中编程相关值。 最多可以8个轴	PRESETON(X,10,Y,4.5)		
PRIO	在处理中断时设置优先级的关键字					
PROC	一个程序的第一个指令			程序段号－PROC－名称		
PTP	点到点运动				m	49
PUTFTOC	刀具精补偿，用于平行修整					
PUTFTOCF	PutFineToolCorrectionFunctionDependant: 刀具精补偿，与一个用FCtDEF确定的功能有关，用于平行修整					
PW	点－加权	实数，无符号			S	
QECLRNOF	象限缺陷补偿学习“关”					
QECLRNON	象限缺陷补偿学习“开”					
QU	快速辅助功能输出					
R...	计算参数，自软件版本SW5起： 也作为可设定的地址名称，并带序号扩展	±0.0000001, …, 9999 9999	R参数个数可以通过MD设定	R10=3 ；R参数赋值. X=R10 ；轴值 R[R10]=6 ；间接编程		
RDISABLE	读入禁止					
READ	在所说明的文件中读入一个或者多个行，并且在数组中存放所读入的信息。		信息为字符串			
READAL	读报警		报警按上升的号查找。			

REAL	数据类型：浮点变量，带符号	符合处理器的64位浮点格式				
REDEF	机床数据设定，NC语言单元和系统变量，在它们的用户组中显示					
RELEASE	加工轴使能		可以编程多个轴			
REP	关键字，用同一个值初始化一个数组的所有单元					
REPEAT	重复一个程序循环		一直进行，直至(UNTIL)一个条件满足			
REPEATB	重复一个程序行		nnn-次			
REPOSA	所有轴再次线性返回运行到轮廓				s	2
REPOSH	以半圆再次返回运行到轮廓				s	2
REPOSHA	所有轴再次返回运行到轮廓；几何轴以半圆				s	2
REPOSL	再次线性返回运行到轮廓				s	2
REPOSQ	以1/4个圆再次返回运行到轮廓				s	2
REPOSQA	所有轴再次线性返回运行到轮廓；几何轴以1/4个圆				s	2
RESET	复位工艺循环		可以编程一个或者多个IDs			
RET	子程序结束		代替M17使用，没有功能输出到PLC	RET		
RINDEX	确定输入端字符串中一个符号的索引	0, ..., INT	字符串：1. 参数，符号：2. 参数			
RMB	再次返回到程序段起始点				m	26
RME	再次返回到程序段结束点				m	26
RMI [1]	再次返回到中断点				m	26

RMN	再次运行到下一个轨迹点				m	26
RND	轮廓角倒圆	实数， 无符号		RND=...	s	
RNDM	模态倒圆	实数， 无符号		RNDM=... RNDM=0: M. V. 关断	m	
ROT	可编程旋转	围绕第一个几何轴旋转 -180° .. 180° 2. 几何轴： -89.999°, ..., 90° 3. 几何轴： -180° .. 180°		ROT X... Y... Z... ROT RPL= ；独立的程序段	s	3
ROTS	可编程的框架旋转，带立体角			ROT X... Y... ROT Z... X... ROT Y... Z... ；独立的 ROT RPL= 程序段	s	3
ROUND	园整逗号后的数据	实数				
RP	极半径	实数			m,s [3]	
RPL	平面中旋转	实数， 无符号			S	
RT	用于存取框架数据的参数：旋转					
S	主轴转速或者（在G4,G96/G961时）其它含义	0.1 ... 99999999.9	主轴转速，单位转/分钟 G4:以主轴转速表示的停留时间 G96/G961:切削速度，单位米/分钟	S...: 转速，用于主主轴 S1...: 转速，用于主轴1	m,s	
SAVE	在子程序调用时保护信息		保护：所有模态G功能和当前框架			
SBLOF	抑制单段		后面的程序段以单段执行，如同一个程序段。			
SBLON	取消单个程序段抑制					
SC	用于存取框架数据的参数：刻度					
SCALE	可编程的比例（刻度）			SCALE X... Y... Z... ；独立的程序段	s	3
SD	样条度	整数， 无符号			S	
SEFORM	在步距编辑器中的结构指令，用于由此生成步距图，用于高级HMI		在步距编辑器中处理	SEFORM(<段名>, <平面>, <图标>)		

SET	关键字，用列表值初始化一个数组的所有单元					
SETAL	设置报警					
SETDNO	重新设置刀具（T）的D号以及其刀沿					
SETINT	确定在出现一个NCK输入端时应该激活哪一个中断程序		处理脉冲沿0→1			
SETM	设置一个/多个标号，用于通道协调		在自身通道中的加工不受影响。			
SETMS	转换回到机床数据中确定的主主轴					
SETMS(n)	主轴n应该作为主主轴					
SETPIECE	考虑用于所有刀具的数量，它们分配到主轴		没有主轴号：适用于主主轴			
SF	起始点偏置，用于螺纹切削	0.0000, ..., 359.999°			m	
SIN	正弦（三角函数）	实数				
SOFT	限制冲击的轨迹加速度				m	21
SOFTA	接通限制冲击的轴加速度，用于编程的轴					
SON [6]	步冲“开”				m	35
SONS [6]	步冲“开”，以IPO节拍				m	35
SPATH [1]	FGROUP轴的轨迹基准是弧长				m	45
SPCOF	主主轴或者主轴从转速控制转换到位置控制			SPCON SPCON (n)		
SPCON	主主轴或者主轴(n)从位置控制转换到转速控制			SPCON SPCON (n)		
SPIF1 [1,6]	快速NCK输入端/输出端，用于冲压/步冲，字节1		参见 /FB/, N4: 冲压和步冲		m	38
SPIF2 [6]	快速NCK输入端/输出端，用于冲压/步冲，字节2		参见 /FB/, N4: 冲压和步冲		m	38
SPLINE-PATH	确定样条连接		max. 8个轴			
SPOF [1,6]	冲程“关”，冲压，步冲“关”				m	35
SPN [6]	每个程序段分段距离个数	整数：			s	
SPP [6]	一个分段距离长度	整数			m	
SPOS	主轴位置			SPOS=10 或者 SPOS[n]=10	m	
SPOSA	主轴位置超过程序段界限			SPOSA=5 或者 SPOSA[n]=5	m	
SQRT	平方根（算术函数）	实数				
SR	摆动退回位移，用于同步动作	实数， 无符号			S	

SRA	在外部输入端轴向（火花放电退回）摆动退回位移，用于同步动作			SRA[Y]=0.2	m	
ST	摆动修光时间（火花放电时间）用于同步动作	实数，无符号			S	
STA	摆动修光时间轴向（轴向火花放电时间）用于同步动作				m	
START	从运行的程序中，在几个通道中同时启动所选择的程序		对自身的通道无效			
STAT	铰接位置	整数			s	
STARTFIFO[1]	执行加工；与此并行补足进刀缓存器				m	4
STOPFIFO	停止加工；补足进刀运行缓存器，直至STARTFIFO识别进刀缓存器已满或者程序结束				m	4
STOPRE	进刀停止，直至所有预备的程序段由主运行加工完毕					
STOPREOF	取消进刀停止					
STRING	数据类型：字符串	最大200个字符				
STRLEN	确定一个字符串的长度	INT				
SUBSTR	确定输入端字符串中一个符号的索引	实数	字符串：1. 参数，符号：2. 参数			
SUPA	抑制当前零点偏移		包括可编程偏移，手轮偏移，(DRF)，外部零点偏移和预置偏移		s	9
SYNFCT	计算一个多项式，取决于运动同步动作中的一个条件	VAR REAL				
SYNR	同步读出变量，也就是说在加工时					
SYNRW	同步读写变量，也就是说在加工时					
SYNW	同步写出变量，也就是说在加工时					
T	刀具调用（仅在机床数据中确定时才更换；否则必需M6指令）	1 ... 32 000	通过T号或者通过刀具名调用	比如T3 或者 T=3 比如 T="BOHRER"		
TAN	正切（三角函数）	Real				
TANG	由两个所给定的引导轴确定切线，用于跟随运行					
TANGOF	切线跟随运行“关”					

TANGON	切线跟随运行“开”					
TCARR	要求刀架（号“m”）	整数	m=0:撤销选择有效的WZ刀架	TCARR=1		
TCOABS [1]	从当前的刀具方向确定刀具长度分量		重新调整后必须进行，比如		M...	42
TCOFR	从当前框架的方向确定刀具长度分量		通过手动设定		m	42
TCOFRX	在选择刀具时确定一个有效框架的刀具方向，刀具指向X方向		刀具垂直于斜面		m	42
TCOFRY	在选择刀具时确定一个有效框架的刀具方向，刀具指向Y方向		刀具垂直于斜面		m	42
TCOFRZ	在选择刀具时确定一个有效框架的刀具方向，刀具指向Z方向		刀具垂直于斜面		m	42
TILT [5]	侧向角	实数			m	
TMOF	撤销选择刀具监控		T-Nr. 只有当这个号的刀具没有效时，才必需T号。	TMOF (T-号)		
TMON	选择刀具监控		T号＝0： 关闭所有刀具的监控	TMON (T-号)		
TO	表示FOR计数循环中的终点值					
TOFFOF	复位在线刀具长度补偿					
TOFFON	激活在线刀具长度补偿		说明一个三维的补偿方向	TOFFON (Z, 25) 带补偿方向Z偏移值25		
TOFRAME	设置当前的可编程的框架到刀具坐标系统		在刀具方向旋转框架		m	53
TOFRAMEX	X轴平行于刀具方向，辅助轴Y,Z				m	53
TOFRAMEY	Y轴平行于刀具方向，辅助轴Z,X				m	53
TOFRAMEZ	Z轴平行于刀具方向，辅助轴X,Y				m	53
TOFROF	框架在刀具方向旋转“关”				m	53
TOFROT	Z轴平行于刀具方向		框架旋转“开”，编程框架的旋转部分		m	53
TOFROTX	X轴平行于刀具方向				m	53
TOFROTY	Y轴平行于刀具方向				m	53
TOFROTZ	Z轴平行于刀具方向				m	53
TOLOWER	一个字符串的字母转换成小写字母					
TOWSTD	基本设定值，用于刀具长度补偿		算入刀具磨损		m	56
TOWBCS	基本坐标系BKS中的磨损值				m	56
TOWKCS	在运动转换时刀具头在坐标系中的磨损值（区别于刀具旋转的MKS）				m	56
TOWMCS	机床坐标系MKS中的磨损值				m	56

TOWTCS	刀具坐标系中的磨损值（刀盘基准点T位于刀具夹持装置中）				m	56
TOWWCS	工件坐标系WKS中磨损值				m	56
TOUPPER	一个字符串的字母转换成大写字母					
TR	用于存取框架数据的参数：平移					
TRAANG	倾斜轴转换		每个通道可以设定多个转换			
TRACEOF	圆弧格式测试：数值传送“关”					
TRACEON	圆弧格式测试：数值传送“开”					
TRACON	级联转换					
TRACYL	圆柱：表面转换		参见 TRAANG			
TRAFOOF	关闭转换			TRAFOOF( )		
TRAILOF	轴同步联动“关”					
TRAILON	轴同步联动“开”					
TRANS	可编程的偏移			TRANS X. Y. Z.；独立的程序段	s	3
TRANSMIT	极坐标转换		参见 TRAANG			
TRAORI	4-,5-轴转换，转换类		激活约定的方向转换	转换类 TRAORI(1,X,Y,Z)		
TRUE	逻辑常量：真	BOOL	可以通过整数常量1代替			
TRUNC	去除小数点后位数	实数				
TU	轴交角	整数		TU=2	s	
TURN	螺旋线圈数	0, ..., 999			s	
UNLOCK	使能带ID的同步动作（继续工艺循环）					
UNTIL	结束一个REPEAT循环的条件					
UPATH	FGROUP轴的轨迹基准是曲线参数				m	45
VAR	关键字：参数转让方式		用VAR：由基准调用			
WAITC	等待，直至轴/主轴的耦合程序段转换准则满足为止		最多2个轴/主轴能够编程	WAITC(1,1,2)		
WAITM	等待所说明通道中的标记；前面的程序段用准停结束			WAITM(1,1,2)		
WAITMC	等待所说明通道中的标记；只有当其它的通道还没有达到该标记时才准停			WAITMC(1,1,2)		

WAITP	等待至运行结束			WAITP(X) ；独立的程序段		
WAITS	等待到达主轴位置			WAITS (主主轴) WAITS (n,n,n)		
WALIMOF	工作区域限制“关”			；独立的程序段	m	28
WALIMON[1]	工作区域限制“开”			；独立的程序段	m	28
WHILE	WHILE程序循环开始		结束： ENDWHILE			
WRITE	程序段写入到文件系统。在所说明文件的结束处加上一个程序段		这些程序段在M30之后插入			
X	轴	实数			m,s [3]	
XOR	逻辑“异－或”					
Y	轴	实数			m,s [3]	
Z	轴	实数			m,s [3]	

图例说明：

[1] 程序开始时的基本设定（指系统供货时没有做其它的编程）。

[2] 组的编号与章节11.3中表格“指令概述”中一致。

[3] 绝对终点：模态方式；增量终点：程序段方式；否则模态方式/程序段方式取决于句法确定的G功能。

[4] IPO参数（增量）作为圆心。使用AC它们可以绝对编程。其它意义时（比如螺距）拒绝地址修改。

[5] 关键字不适用于 SINUMERIK FM-NC/810D。

[6] 关键字不适用于 SINUMERIK FM-NC/810D/NCU571。

[7] 关键字不适用于 SINUMERIK 810D。

[8] OEM用户可以另外再提出两个插补方式。.名称可以由OEM用户改变。

[9] 关键字仅适用于 SINUMERIK FM-NC。

[10] 对于该功能扩展的地址写入方式不允许。

15.2 系统变量清单



文献说明：

自软件版本**SW7.1**起，系统变量在手册：
SINUMERIK 840D/840Di/810D“系统变量清单”
中列出。

■

附录

A 缩略符

A	输出端
AS	可编程控制系统
ASCII	美国信息交换标准代码信息交换美国代码标准
ASIC	专用集成电路应用器件用户开关回路
ASUP	异步子程序
AV	工作准备部分
AWL	指令表
BA	工作方式
BAG	工作方式组
BB	运行准备
BuB, B&B	操作与编程
BCD	二－十进制二进制编码十进制数
BHG	操作设备
BIN	二进制文件
BIOS	基本输入输出系统
BKS	基本坐标系统
BOF	操作界面
BOT	引导文件SIMODRIVE 611D引导文件
BT	操作面板
BTSS	操作面板接口
CAD	计算机辅助设计



CAM 计算机辅助制造

CNC 计算机数字控制计算机数字控制系统

COM 通讯

CP 通讯处理器

CPU 中央处理单元中央处理单元

CR 回车

CRT 阴极射线管阴极射线管

CSB 中央服务板PLC 组件

CTS 清除发送通过串行数字接口通知准备就绪

CUTOM 刀具半径补偿刀具半径补偿

DAU 数字模拟转换器

DB PLC中数据块

DBB PLC中数据块字节

DBW PLC中数据块字

DBX PLC中数据块位

DC 直接控制在一转内回转轴以最短距离移动到绝对位置

DCD 载波检测

DDE 动态数据交换

DEE 数据结束调试

DIN 德国工业标准

DIO 数据输入/输出数据传送显示

DIR 路径路径



DLL	动态连接程序库
DOE	数据传送设备
DOS	磁盘操作系统
DPM	双端口存储器
DPR	双端口存储器
DRAM	动态随机存取存储器
DRF	DRF功能直接测量功能（手轮）
DRY	空运行空运行
DSB	单段译码单段译码
DW	数据字
E	输入端
E/A	输入/输出
E/R	SIMODRIVE 611(D)电源/回馈模块
EIA-Code	专门穿孔带编码，每个字符的穿孔数始终为奇数
ENC	编码器实际值编码器
EPROM	可擦除可编程只读存储器
ERROR	打印机错误
FB	功能块
FBS	超薄显示屏
FC	功能调用PLC中功能块
FDB	生产数据库
FDD	软盘驱动器

FEPROM	闪存EPROM读写存储器
FIFO	先进先出存储器，工作无需地址说明，数据按存储的顺序读入
FIPO	精插补
FM	功能模块
FM-NC	功能模块数字控制
FPU	浮点单位浮点单位
FRA	框架模块
FRAME	数据块
FRK	铣削半径补偿
FST	进给停止进给停止
FUP	功能图（PLC编程方法）
GP	主程序
GUD	全局用户数据全局用户数据
HD	硬盘硬盘



HEX 十六进制数代号

HiFu 辅助功能

HMS 高分辨率测量系统

HSA 主轴驱动

HW 硬件

IBN 开机调试

IF 驱动模块脉冲使能

IK (GD) 隐含通讯（全局数据）

IKA 可插补补偿可插补补偿

IM 接口模块接口模块

IMR 接收方接口模块接收方接口模块

IMS 发送方接口模块发送方接口模块

INC 增量方式增量方式

INI 初始化数据初始化数据

IPO 插补器

ISA 国际标准体系

ISO 国际标准组织

ISO-Code 专门穿孔带编码，每个字符的穿孔数始终为偶数

JOG 手动工作方式手动工作方式

K1 ..K4 通道1到通道4

K-Bus	通讯总线
KD	坐标旋转
KOP	功能图（PLC编程方法）
K_v	回路放大系数
$K_Ü$	传动比
LCD	液晶显示液晶显示
LED	发光二极管发光二极管
LF	线路馈电
LMS	位置测量系统
LR	位置调节器
LUD	局部用户数据
MB	兆字节
MD	机床数据
MDA	手动输入，自动运行手动输入，自动运行
MK	测量回路
MKS	机床坐标系
MLFB	机器可识别产品符
MMC	人机通讯操作、编程和模拟运行界面
MPF	主程序文件NC零件程序（主程序）
MPI	多端口接口多端口接口



MS-	微软（软件制造商）
MSTT	机床控制面板
NC	数字控制数字控制
NCK	数字控制核心数字控制核心
NCU	数字控制单元NCK硬件单元
NRK	NCK操作系统名称
NST	接口信号
NURBS	非一致性数理B样条
NV	零点偏移
OB	PLC中组织块
OEM	原设备制造商
OP	操作面板操作面板
OPI	操作面板接口操作面板接口
OPT	选件选件
OSI	开放式互联系统计算机通讯标准
P-Bus	外设总线
PC	个人计算机
PCIN	与控制系统进行数据更换的软件名称
PCMCIA	个人计算机存储卡国际协会存储器插卡标准
PG	编程器
PLC	可编程逻辑控制器可编程逻辑控制器
POS	定位

RAM	随机存取存储器程序存储器，可读写
REF	回参考点运行
REPOS	再定位功能
RISC	简化的计算机指令系统处理器类型，具有较小的指令组、快速的指令处理能力
ROV	快速倍率快速倍率
RPA	R参数有效存储器范围 NCK中用于R参数号
RPY	旋转定位移动一种坐标系旋转方式
RTS	请求发送开启发送方，控制信号自串行数据接口
SBL	单段单段
SD	设定数据
SDB	系统数据块
SEA	设定数据有效设定数据符号（文件类型）
SFB	系统功能块
SFC	系统功能调用
SK	软键
SKP	程序段跳跃程序段跳跃
SM	步进电机

SPF	子程序文件子程序
SPS	存储器可编程控制
SRAM	静态存储器（缓存）
SRK	刀尖补偿
SSFK	丝杠螺距误差补偿
SSI	串行同步接口串行同步接口
SW	软件
SYF	系统文件系统文件
TEA	测试数据有效机床数据标志
TO	刀具补偿刀具补偿
TOA	刀具补偿有效刀具补偿符号（文件类型）
TRANSMIT	铣削转换为车削铣削加工中车床坐标系换算
UFR	用户框架零点偏移
UP	子程序
VSA	进给驱动
V.24	串行接口（DEE和DUE之间数据交换定义）
WKS	工件坐标系
WKZ	刀具
WLK	刀具长度补偿

WOP	现场编程
WDP	工件目录工件目录
WRK	刀具半径补偿
WZK	刀具补偿
WZW	换刀
ZOA	零点偏移有效零点偏移数据符号（文件类型）
μC	微米级控制器

B 术语

按照字母顺序给出术语说明。说明文字中出现的术语有单独的出处说明，在此用->表示。

A

A样条

Akima样条始终以编程的支点进行切线移动（三项式）。

B

B 样条

在B样条中，编程的位置不是支点，而仅仅是“控制点”。产生的曲线不是直接经过控制点，而仅仅是在它们的附近（可选择一级、二级或三级多项式）。

C

C 样条

C样条最出名，是一种最常用的样条。支点处以切线过渡，弯曲平缓。使用三级多项式。

C 轴

围绕C轴产生一个受控的旋转运动，并用工件主轴定位。

CNC

->NC

CNC 标准语言

CNC标准语言提供：用户变量，预定义用户变量，系统变量，间接编程，运算功能和三角函数功能，比较运算和逻辑运算，程序跳转和分支，程序协调（SINUMERIK 840D），宏指令

CNC编程语言

CNC编程语言的基础是DIN66025,带高级语言扩展。此外，CNC高级语言和编程语言允许使用宏指令定义（单个指令的汇编）。

COM

NC控制系统部件，用于执行和和协调通讯。

CPU

中央处理单元，->存储器可编程控制器

D

DRF

DRF功能NC功能，在自动方式下利用电子手轮产生增量式零点偏移。



H

HIGHSTEP AS300/AS400系统中PLC所有编程方法的汇编。

J

Jog方式 控制系统的一种运行方式（调试运行）：在Jog运行方式下，机床可以进行调试。各个进给轴和主轴可以通过方向键点动运行。在Jog手动运行方式中还有其它的一些功能，如回参考点运行，再定位以及预设定（设定实际值）。

K

$K_Ü$ 传动比

K_v 回路放大系数，调节回路中可调节的物理量。

M

MDA 控制系统的一种运行方式：手动输入，自动运行在MDA方式下，可以输入单个程序段或者几个程序段，它们与主程序或者子程序无关，使用NC启动键可以立即执行。

N

NC 数字控制NC控制系统中包含机床控制系统的所有部件。 -> NCK, -> PLC, -> MMC/HMI -> COM.
说明：对于SINUMERIK 840D或者FM-NC控制系统，
CNC控制系统应改为：计算机数控系统。

NCK 数字控制核心NC控制系统部件，执行零件程序，并控制机床的运动过程。

NRK 数字机器人内核（NCK的运行系统）

NURBS 系统内部的运动控制和轨迹插补根据NURBS(Non Uniform Rational B-Splines)进行。这样，在系统内部所有插补均有相同的方法（SINUMERIK 840D）。

O

OEM

SINUMERIK 840D
给机床制造商提供各种不同应用的使用空间（OEM应用），制造商可以自己设计操作界面或者在系统中开发专用的应用功能。

P

PG

编程器

PLC

可编程逻辑控制器存储器可编程控制NC控制系统部件：用于执行机床控制逻辑的转接控制。

PLC-编程

PLC用软件STEP 7编程。编程软件STEP7基于WINDOWS标准软件、在STEP5编程功能的基础上发展的。

PLC-编程存储器

SINUMERIK 840D:在PLC用户存储器中，
PLC用户程序和用户数据与PLC主程序一起存储。PLC用户存储器可以通过存储器扩展至96K字节。

R

REPOS

1. 通过操作返回轮廓。使用Repos功能可以通过方向键再次回到中断点。
2. 通过程序返回轮廓。通过编程指令，可以选择几种不同的返回：返回到中断点，返回到程序段起始点，返回到程序段终点，返回到程序段起始点和中断点之间的一个轨迹点。

R参数

计算参数，可以由零件程序编程人员在程序中进行任意设定或者询问。

S

S7-300 总线

S7-300总线是一个串连数据总线，
通过该数据总线模块可以相互进行通讯，同时总线自身提供电源。模块之间的联系通过总线连接器建立。

S7配置

S7配置是一个工具，用此工具可以给模块设定参数。使用S7配置可以在编程器上设定CPU和外设模块的各个参数块。
这些参数传送到CPU中。

SPS 存储器可编程控制系统

上电 关机后再次开机。

丝杠螺距误差补偿 滚珠丝杠在进给时产生机械误差，由控制系统通过存储的误差测量值进行补偿。

中断程序 中断程序是专门的子程序，它们可以通过加工过程中的外部事件（外部信号）启动。加工过程中零件程序的程序段被中断，进给轴的中断位置被自动存储。

中间程序段 带刀具补偿（G41/G42）的加工过程可以由一定数量的中间程序段（在补偿级的程序段，没有轴运动）中断，这样刀具补偿还可以进行正确地计算。先于控制系统读出所允许的中间程序段数量，可以通过系统参数设定。

串行接口RS232 在数据进行输入/输出时，

- 在MMC模块MMC100上有一个串行接口RS232，
- 在MMC模块MMC101和MMC102中有两个RS232接口。

通过该接口可以装载和保护加工程序以及制造商和用户数据。

主程序 用序号或者名称标志的零件程序，在主程序中可以调用其它的主程序、子程序或者循环。

主程序段 通过“：”引导的程序段，包含在零件程序中启动工作流程所需要的所有数据。

主轴 主轴功能分为两种功率级别：

1. 主轴：转速控制或者位置控制的主轴运行，数字式（SINUMERIK840D）
2. 辅助主轴：转速控制的主轴驱动，功能包“辅助主轴”比如用于驱动的刀具。

仿真器模块 仿真器模块是一种模块，

- 通过操作部件可以模拟数字输入量，
- 显示数字输出量

保护区 在加工区之内的一个三维空间，刀尖不可以进入此区域。

信息

零件程序中可编程的所有信息，以及系统可识别的报警均在操作面板上显示，带日期和时间，并有相应的清除标准符号。
报警和信息单独显示。

倍率（Override）

可以手动或者编程进行工作，允许操作人员覆盖编程的进给或者转速，使加工速度与具体的工件和材料相适应。

倒圆轴

倒圆轴指工件或者刀具旋转到一个分度头给定的角度位置。到达分度头刻度后，倒圆轴“到达位置”。

全局主程序/子程序

在一个目录下每个全局主程序/子程序只可以出现一次，不可以在不同目录下有不同内容的程序具有相同的程序名。

公制测量系统

单位均为公制：比如长度为毫米、米。

关键字

有确定写法的字，它们在编程语言中具有所定义的含义。

准停

使用编程的准停指令，可以准确地、有时必须较慢地回到程序段中所设定的位置。为了减少准停时的逼近时间，对于快速移动和进给需定义准停界限。

准停界限

如果所有的轨迹轴均到达准停界限，则控制系统会认为已经精确到达目标。进行零件程序的程序段转换。

几何尺寸

工件在工件坐标系中的描述。

几何轴

几何轴用于描述工件坐标系中2维或者3维的尺寸。

刀具

机床中进行加工的部件，诸如车刀、铣刀、钻头、激光…

刀具半径补偿

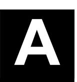
为了可以直接编程一个所要求的工件轮廓，控制系统必须考虑所使用刀具的半径，与编程的轮廓等距离运行。(G41/G42).

刀具补偿

在程序段中编程一个T功能（5位整数）可以选择刀具。每个T号可以最多有9个刀沿（D地址）。控制系统中所管理的刀具数量可以通过设计进行修改。

刀尖半径补偿

在编程一个轮廓时，往往从刀具的尖端计算。但是，这在实际加工过程中并不可以实现，因为所使用的刀具会有一个弯曲半径，系统必须要考虑这个值。在此计算的加工点就位于其中心点，距离为半径的长度。



初始化文件	对应于每个工件可以编制一个初始化文件。在初始化文件中可以编制不同变量的赋值指令，它们仅适用于一个工件。
初始化模块	初始化模块是专用的程序模块。它包含在程序处理之前须执行的赋值。初始化模块主要用于初始化预定义的数据或者全局用户数据。
剩磁	剩磁是指数据块中的数据区以及定时器、计数器和标志位在新启动时，或者在掉电时不会丢失。
加工	系统操作区。
加工空间	用加工空间定义一个三维空间，在此空间内刀尖可以移动。 参见->保护空间
加工轴	在机床中表示实际存在的轴。
加工通道	通过通道结构可以进行并行处理，缩短辅助时间，比如在装载的同时可以进行加工。在此，一个CNC通道可以看作为一个独力的CNC控制系统，可以译码、程序段预处理并进行插补。
加速度，带冲击限制	为了在机床上获得优化的加速性能，同时又要保护机械部分，在加工程序中可以在突变式加速度和平缓式加速度之间进行转换。
参数	1. **S7-300:**可以分为两种参数类型： - STEP 7 指令参数 STEP 7 指令的参数就是待加工的操作数地址或者常数。 - 一个参数块的参数 一个参数块的参数确定一个模块的性能。 2. **840D**: - 系统操作区。 - 计算参数，可以由零件程序的编程人员在程序中任意设定或者询问。
参考点	机床中的一点，加工轴的测量系统以此为基准。
反比时间进给	在SINUMERIK840D中，轴运动不是编程进给速度，而是编程一个程序段轨迹位移所需要的时间（G93）。



变量定义	定义变量时包括确定数据类型和变量名。使用该变量名，也就是调用该变量值。
可插补补偿	利用插补补偿功能可以补偿生产过程所决定的丝杠螺距误差和测量系统误差。
可编程的工作区域限制	刀具的运行区域限制到一个通过编程限制的区域范围。
可编程的框架	使用编程的框架可以在零件程序加工过程中，动态地定义新的坐标系原点。根据当前的原点，利用一个新框架和附加的确定值，与绝对的确定值加以区分。
同步	零件程序中的指令，用于协调同一加工地点时不同通道中的加工过程。
同步动作	1. 辅助功能输出 在工件加工期间，可以把工艺功能（辅助功能）从CNC程序中输出到PLC中。通过辅助功能可以控制机床的附加设备，比如顶尖套筒，夹持器，卡盘等等。 2. 快速辅助功能输出 对于时间较紧的开关功能，可以减少辅助功能的应答时间，避免加工过程不必要的停顿。
同步轴	同步轴运行时间与几何轴相同。
名称	根据DIN 66025标准，字需要补充变量名（计算变量，系统变量和用户变量）、子程序名、关键字名和带多个地址字母的字。这些补充的字在意义上与构成程序段的字一样。名称必须意义明确。同一个名称不可以用于不同的对象。
回参考点	如果所使用的位移测量系统没有绝对值编码器，则必须要回参考点运行，从而保证测量系统所提供的实际值与机床坐标值相一致。
回机床固定点	返回到预定义的机床固定点。
回转轴	回转轴指工件或者刀具旋转到一个给定的角度位置。



回转轴无限旋转 根据具体的应用场合，回转轴可以旋转小于360度，或者在两个方向无限旋转。无限旋转的回转轴，比如可以用于非圆加工、磨削加工和绕线加工。

圆弧插补 在轮廓上两个固定点之间，刀具以给定的进给量按圆弧运行，从而加工出工件。

地址 地址是一个确定的运算数或者运算范围的标志，比如输入、输出等等。

坐标系 参见机床坐标系、工件坐标系。

型材导轨 型材导轨用于固定S7-300的模块。

基准坐标系 是一个直角坐标系，它通过转换到机床坐标系而形成。
在零件程序中使用基准坐标系编程的轴名称。如果没有坐标系转换，则它平行于机床坐标系。不同点在于轴名称。

基准轴 计算补偿值时必须考虑该轴的给定值或者实际值，这个轴就称为基准轴。

增量尺寸 也称为相对尺寸：表示一个进给轴待运行的行程和方向，以已经到达的点为基准。参见->绝对尺寸。

增量方式 通过相对尺寸说明加工行程。相对尺寸可以作为设定数据存储，或者通过相应的增量键10、100、1000和10000进行选取。

备份 存储器内容存储到外部存储设备中。

备份存储器 备份存储器保证存储器存储区的缓冲状态下工作，
CPU没有缓存电池。定时器、计数器、标志和数据字节数可以设定参数并缓存。

备份电池 利用备份电池保证在电网掉电时，用户程序可以安全地存放在CPU中，并且确定的数据区以及剩余的标志位、定时器和计数器可以保持。

外设模块 用外设模块建立CPU和过程之间的联系。外设模块是：
- 数字量输入/输出模块
- 模拟量输入/输出模块
- 仿真器模块

外部零点偏移
由PLC给定的零点偏移。

多端口接口
多端点接口（MPI）是一个9芯的D-Sub接口。
通过多端口接口可以连接一系列设备，相互可以进行通讯：

- 编程器
- 操作与监控系统
- 其它可编程控制器

CPU的“多端口接口MPI”参数组包含有各个参数，用这些参数可以确定多端口接口的性能。

多项式插补
用多项式插补功能可以产生不同的曲线，比如线性函数、抛物线函数和幂函数（SINUMERIK 840D）。

子程序
一个子程序的连续指令，它们可以通过设定不同的参数反复调用。子程序从主程序中调用。没有授权的读取和显示会被子程序禁止。循环是子程序的一种。

存储器可编程控制系统
存储器可编程的控制系统（SPS）是电子控制系统，它们的功能以程序的形式存储到控制器中。因此，控制器的结构和布线与控制系统的功能无关。存储器可编程的控制系统具有计算机的结构，它由带存储器的CPU（中央模块）、输入/输出模块和内部总线系统构成。外设和编程语言以控制技术为准。

存取权限
CNC程序块和数据通过一个7级存取权限进行保护。

- 三个口令字，分别用于系统生产厂家、机床制造商和用户，以及
- 4个钥匙开关位置，可以由PLC进行利用

存档
读出文件或目录，存储到外部存储器设备中。

安全功能
系统中所具有的安全监控功能，通过安全监控功能数控系统中以及PLC和机床中的故障均可以尽早地予以识别，从而排除一切对工件、刀具或者机床可能造成的危害。在故障发生时，加工过程会中断，驱动停止，故障原因被存储并作为报警显示。同时通知PLC数控系统有一报警。

宏指令
一个指令名称下汇编一串指令。在程序中，该指令名就代表这一串汇编的指令。

定位轴	在机床中执行辅助运动的轴（比如刀库，托盘运输）。定位轴不与轨迹轴进行插补。
定向主轴准停	比如主轴在一给定角度位置停止，从而可以在某一固定位置进行其它的加工工作。
定向刀具退回	RETTOOL:当加工过程被停止时（比如刀具折断），刀具可以根据编程指令按照事先给定的方向后撤一段距离。
尺寸系统：公制和英制	在加工程序中，位置值和螺距值可以用英制编程。控制器设定一个基准系统，它与编程的尺寸系统（G70/G71）无关。
工件	机床待加工的零件。
工件坐标系	工件坐标系中有工件零点。在工件坐标系中编程时，尺寸和方向以工件坐标系为基准。
工件轮廓	待加工工件的给定轮廓。
工件零点	工件零点构成了工件坐标系的原点。它由与机床零点的距离定义。
工作区域限制	除行程开关之外，还可以使用工作区域限制功能对进给轴的行程范围进行限制。对于每个进给轴，可以使用两个数值对保护加工区进行设定。
工作存储器	工作存储器是一个RAM存储器，在程序加工期间处理器可以对用户程序进行存取。
工作方式	SINUMERIK控制系统的运行过程控制方式。它们是下面几种工作方式：Jog（手动运行方式），MDA（手动输入，自动运行方式），自动方式。
工作方式组	在某一时间所有的进给轴/主轴均精确地列入到某一个通道，每一个通道均列入一个工作方式组。 同一个工作方式组中的通道均有相同的工作方式。

工具

一个工具是指用于输入和修改参数组参数的软件工具。工具如：

- S7配置
- S7-TOP
- S7-信息

异步子程序

指可以通过一个中断信号（比如信号“快速NC输入”）启动的、与当前程序状态异步（无关）的子程序。

引导

上电后装载系统程序。

循环

受保护的子程序，用于执行工件中反复出现的加工过程。

循环辅助

在“程序”操作区菜单“循环”下，列出所有供使用的循环清单。选择了所要求的加工循环后，屏幕上会显示参数赋值指令中必须设定的参数。

快速提刀

当中断加工时，可以通过CNC加工程序引入一个动作，使刀具从所加工的工件轮廓快速离开。此外还可以设定退刀的角度和位移的参数。在快速提刀以后可以另外执行一个中断程序。(SINUMERIK 840D).

快速数字输入/输出

通过数字输入端可以启动快速CNC程序（中断程序）。通过数字输出端可以释放快速的、程序控制的开关功能。(SINUMERIK 840D).

快速移动

轴运行最快速度比如，当刀具由静止状态运行到工件轮廓或者由工件轮廓返回时使用快速移动速度。

总线连接器

总线连接器是S7-300的附件，它与外设模块一起提供。
通过总线连接器，S7-300总线可以从CPU
或者一个外设模块扩展到其相邻的外设模块。

成品轮廓

成品工件的轮廓。参见->毛坯件。

报警

所有的信息和报警均在操作面板上显示其文本，带日期和时间，并有相应的清除标准符号。报警和信息单独显示。

1. 零件程序中报警和信息
   报警和信息可以直接从零件程序中以文字形式显示。
2. PLC的报警和信息
   机床的报警和信息可以从PLC程序中以文字形式显示。在此无需另外的功能块软件包。

接地

接地是指在设备中连接到一起的所有无源器件，它们即使在出现故障时也不会有危险电压。

插补器

NCK的逻辑单元，根据零件程序中目标位置的参数确定进给轴待运行的中间值。

操作

系统操作区。

操作界面

操作界面（BOF）是CNC控制系统的显示形式，带屏幕。它带有八个水平软键和八个垂直软键。

攻丝，不带补偿衬套

用此功能可以不带补偿衬套攻丝螺纹。通过插补运行，主轴作为回转轴和钻削轴进行螺纹加工，精确地至钻削深度，比如盲孔螺纹（前提条件：主轴作为进给轴运行）。

数字量输入/输出模块

数字量模块用于二进制过程信号的处理。

数据传送程序PCIN

PCIN是一种辅助程序，通过串行接口发送和接收CNC用户数据，比如零件程序、刀具补偿等等。PCIN程序可以在标准工业计算机中MSDOS下运行。

数据块

1. 数据单元，PLC可以对HIGHSTEP程序进行存取。
2. 数据块->NC：数据块包含全局用户数据的数据定义。
   数据可以在定义时直接初始化。

数据字

在数据块中两个字节大小的数据单位。

文本编辑器

编辑器

斜面加工

在工件表面进行钻削和铣削加工，它们不在机床坐标平面，但是可以通过“斜面加工”功能很方便地实现。

机床固定点

在机床中明确定义的点，比如参考点。

机床坐标系

以机床轴为基准的坐标系。

机床控制面板

机床中具有各个操作按键、旋钮开关以及各个显示单元如 LEDs 的控制面板，它们通过PLC对机床进行控制。

机床零点

机床固定点，所有测量系统均可以以此点为出发点。

极坐标

极坐标系指在一个平面中确定一个点的位置，它由到零点的距离与半径矢量和一个轴之间的夹角确定。

极限速度

最大/最小（主轴）速度：通过在机床数据、PLC数据或者设定数据中的规定，可以限制主轴的最大速度。

标准循环

对于经常出现的加工情形，可以使用标准循环：

- 适用于钻削/铣削
- 适用于车削

在“程序”操作区菜单“循环”下，列出所有供使用的循环清单。

选择了所要求的加工循环后，屏幕上会显示参数赋值指令中必须设定的参数。

样条插补

通过样条插补，控制系统可以由理论轮廓上较少的、给定的支点生成一条光滑的曲线。

框架

框架定义一种运算规范，它把一种直角坐标系转换到另一种直角坐标系。一个框架包含几个分量->零点偏移->旋转->刻度->镜像。

模块

模块是指编程和程序执行时所需要的所有文件。

模拟量输入/输出模块

模拟量输入/输出模块用于模拟量过程信号的处理。

模拟量输入模块用来把模拟量测量值转换为CPU可以处理的数字量数值。

模拟量输出模块用来把数字量数值转换为模拟量的调节参数。

比例尺

是构成框架的一个部分，可以改变某个轴的比例尺。

毛坯

毛坯指用于工件加工的原材料。

波特率

数据传送时的速度（位/秒）。

测量回路

- SINUMERIK FM-NC：在缺省情况下，
  进给轴和主轴所必需的测量回路已经集成到控制模块中。总共可以有4个进给轴和主轴，其中最多为2个主轴。
- SINUMERIK 840D:测量传感器在SIMODRIVE 611D
  驱动模块中处理。最大配置可以达到8个进给轴和主轴，其中最多允许5个主轴。

清零

在清零时，CPU中以下的存储器将被清零：

- 工作存储器
- 装载存储器的读写区
- 系统存储器
- 备份存储器

漂移补偿

在CNC轴恒定量运行期间产生一个自动漂移补偿，用于模拟量转速调节(SINUMERIK FM-NC)。

用户号码

如果几个用户通过网络进行通讯，则用户号码表示一个CPU或者编程器的“动作地址”，或者一个其它的智能外设模块的“动作地址”。使用S7工具“S7-配置”给CPU或者编程器赋值一个用户号码。

用户存储器

所有的程序和数据，比如零件程序、子程序、注释、刀具补偿、零点偏移、框架以及通道和程序用户数据均可以存储到共同的CNC用户存储器中。

用户定义变量

用户可以定义用户变量，从而可以在零件程序或者数据块（全局用户数据）中任意使用。一个定义通常含有数据类型和变量名称。
参见->系统变量。

用户程序

可编程控制器S7-300中用户程序用STEP7语言编写。
用户程序为模块化结构，由各个模块构成。
基本的模块类型有：
代码模块：该模块含有STEP7指令。
数据模块：该模块含有用于STEP7程序的常量和变量。

电子手轮

利用电子手轮可以在手动运行状态运行所选择的轴。手轮上刻度线值的大小由步距值确定。

示教

使用示教功能可以编制或者修改零件程序。各个程序段可以通过键盘输入，并可立即运行。通过方向键或者手轮运行的位置也可以存储。附加数据，如G功能、进给率或者M概念可以输入到同一个程序段中。

程序

1. 系统操作区。
2. 到控制系统的连续指令。

程序块

程序块包含零件程序的主程序和子程序。

程序段

零件程序的一个部分，换行后结束。分为主程序段和辅助程序段。

程序段搜索

在进行零件程序测试时或者在中断一个加工后，可以通过程序段搜索功能找到程序中的任意位置，在此位置加工可以启动或者继续。

系统变量

无需编程员的工作，已经存在的变量。它由数据类型和变量名定义，变量名之前有符号**$**。参见用户定义的变量。

系统存储器

系统存储器是CPU中的一个存储器，其内容为：

- 操作系统所需要的数据
- 运算的定时器、计数器和标志位

线性插补

刀具以直线运行到目标点，同时进行工件的加工。

线性轴

与回转轴相反，线性轴指按直线运行的轴。

结构技术

- SINUMERIK FM-NC 排列到SIMATIC S7-300 中CPU 列200毫米宽、有外壳包围的模块外表与SIMATIC S7-300 模块一致。
- SINUMERIK 840D 作为紧凑型模块与变频器系统SIMODRIVE 611D排列在一起。尺寸与一个50毫米宽的SIMODRIVE 611D模块一致。SINUMERIK 840D 模块由NCU 模块和NCU盒组成。

绝对尺寸

进给轴在某一方向上移动说明，表明在当前坐标系中离开零点的距离。参见->增量尺寸。

编程码

编程码是一种字符和字符串，它们在零件程序的编程语言中具有确定的含义（参见编程说明）。

编辑器

利用编辑器可以进行程序/文本/程序段的编辑、修改、合并和插入。

网络

网络指通过连接电缆连接几个S7-300和其它终端设备，比如一台编程器。通过网络进行相连设备之间的数据交换。

翻转

框架的一个部分，定义坐标系按照一定的角度进行旋转。

自动方式

控制系统的运行方式（程序段连续运行，符合DIN标准）：NC系统中的运行方式，这种方式下选择零件程序并连续加工执行。

英制尺寸系统

长度以“英寸”及其导出单位为尺寸的测量系统。

螺旋线插补

螺旋线插补特别适用于利用成形铣刀简单地加工内螺纹和外螺纹，以及铣削润滑槽。在这里螺旋线由两个运动组成：

1. 平面中的回转运动
2. 与此平面垂直的直线运动

补偿值

测量传感器所测得的轴位置与所要求的、编程的轴位置之间的差值。

补偿存储器

控制系统中的一个数据区，刀具补偿数据存储在其中。

补偿表

支点表补偿表给基准轴所选择的位置提供补偿轴的补偿值。

补偿轴

设定值或者实际值可以通过补偿值进行修改的轴。

装载存储器

在SPS的CPU314中，装载存储器就等同于
工作存储器

设定数据

设定数据确定机床的性能，按照系统软件定义的方法在系统中设定。

诊断

1. 系统操作区。
2. 控制系统不仅有自诊断程序，而且还可以进行维修时辅助测试。状态、报警和服务信息。

语言

操作界面的显示文本和系统信息、报警可以有五种语言（磁盘）：
德语，英语，法语，意大利语和西班牙语。
在系统中可以选择并同时安装以上语言中的两种。

象限误差补偿

在象限过渡时，由于在导轨面上出现不同的摩擦而引起的轮廓误差，可以通过象限误差补偿予以消除。象限误差补偿的参数可以通过圆弧形状测试确定。

轨迹控制运行

轨迹控制运行的目的在于：避免在零件程序的程序段结束处轨迹轴产生较大的制动，影响系统、机床以及运行和用户参数值，从而尽可能地以相同的轨迹速度更换到下一个程序段。



轨迹轴	轨迹轴指所有的加工轴，通道由插补器控制，它们可以同时启动、加速、停止直至到达终点。
轨迹进给	轨迹进给影响轨迹轴。表明相关几何轴其进给量的几何量总和。
轨迹速度	最大可编程轨迹速度与进给精度有关。比如精度为0.1毫米，则可编程的最大轨迹速度为1000米/分钟。
转换	在一个直角坐标系中编程，在一个非直角坐标系中加工（比如加工轴作为回转轴）。
轮廓	工件的外部轮廓
轮廓监控	作为轮廓监控的尺寸，滞后量误差控制在一个可定义的公差带之内。比如，当驱动负载过大时就可能产生一个不允许的、过高的滞后量误差。在这种情况下会产生一个报警，从而轴停止运行。
轮廓破坏预先识别	控制系统识别和通报以下的轮廓冲突情形： 1. 轨迹行程短于刀具半径。 2. 内角的宽度小于刀具直径。
软件限位开关	软件限位开关限制一个轴的移动范围，阻止滑枕冲撞硬件限位开关。每个轴可以给定两组数值，它们可以由PLC分别激活。
软键	软键在屏幕上显示，具有对应的区域，可以动态地与当前的操作情形相对应。这些功能键（软键）可以自由分配，它们由软件按照定义的功能进行分配。
轴名称	参见->进给轴命名
轴地址	参见->进给轴命名
辅助功能	在零件程序中，使用辅助功能可以把机床制造商定义的参数传送到PLC中，并释放其所定义功能。
辅助程序段	通过“N”引导的程序段，包含一个加工步骤的信息，比如一个位置说明。
运行范围	线性轴中最大允许的运行范围可以达到±9位。绝对值取决于所选择的输入单位和位置控制单位，以及单位制（英制或者公制）。

返回固定点	机床中可以定义一些固定点，比如刀具更换点、装料点、托盘更换点等等，并可返回。这些点的坐标存储到控制系统中。控制系统控制相关轴运行，如果可能->以快速方式运行。
进给倍率	通过机床控制面板或者PLC可以调节实际速度，并覆盖编程的速度（0－200%）。另外，进给速度也可以在加工程序中，通过一个编程的百分比（1－200%）进行修改。
进给轴	数控系统中的进给轴根据其功能可以分为： • 进给轴可插补的轨迹轴 • 辅助轴不可插补的横向进给和定位轴，具有轴向进给功能。辅助轴不参与加工，比如刀具供料器、刀具库。
进给轴名称	进给轴根据DIN 66217标准中右向旋转直角坐标系命名：X,Y,Z 围绕X, Y, Z旋转的回转轴命名为A, B, C。其它平行的进给轴可以用其它地址字母标识。
连接电缆	连接电缆指预制的或者由用户自己定制的两芯电缆，带两个插头。连接电缆通过多端口接口（MPI）把CPU与编程器或者其它CPU相连。
通道	一个通道是指可以单独处理一个零件程序，而与其它的通道无关。一个通道仅控制其所分配的进给轴和主轴。不同通道的零件程序其加工过程可以通过同步功能进行协调。
通道结构	利用通道结构可以同时/分开加工各个通道的程序。
速度控制	在轴移动时，为了可以使每个较小行程的程序段达到一个可以承受的运行速度，可以使用处理多个程序段的预见功能（->Look Ahead）。
钥匙开关	1. **S7-300**:钥匙开关是CPU的运行方式开关。 钥匙开关的操作通过一个可以插拔的钥匙进行。 2. **840D**:机床控制面板上的钥匙开关有4个位置，它们由控制器的操作系统分配相应的功能。钥匙开关有3个不同颜色的开关，它们可以在所给定的位置插拔。
镜像	使用镜像功能，使加工轮廓相关轴的坐标值符号相反。可以同时对多个轴进行镜像。
间隙补偿	机床在机械方面的间隙补偿，比如滚珠丝杠的反向间隙。对于每个轴，可以分别输入间隙补偿。

零件程序

NC控制系统中的连续指令，它们一起加工出确定的工件。也就是说在一个所提供的毛坯上进行一定的加工。

零件程序管理

零件程序可以按照工件管理。用户存储器的尺寸确定所管理的程序和数据的数量。每个文件（程序和数据）可以命名最多24个字母数据字符的名称。

零点偏移

在一个坐标系中，相对于目前的零点和框架规定一个新的基准点。

1. 可调定：SINUMERIK FM-NC：
   每个CNC进给轴可以选择4个相互独立的零点偏移。
   SINUMERIK 840D:对于每个CNC轴，
   可以设定不同数量的零点偏移。通过G功能可选择的偏移可以选择性地使用。
2. 外部：
   所有确定工件零点位置的偏移可以通过手轮（DRF偏移）或者通过PLC由一个外部零点偏移覆盖。
3. 可编程：
   使用TRANS指令可以给所有的轨迹轴和定位轴编程零点偏移。

预控制，动态

滞后量误差所决定的轮廓误差，几乎可以通过动态的、由加速度决定的预控制消除。由此可以获得一个非常好的加工精度，即使是在轨迹速度很高的情况下。预控制可以通过零件程序根据相应的轴选择或者撤销选择。

预见功能

利用预见功能（Look Ahead），可以通过“预见”
几个可参数化的程序段而获取加工速度的最优化。

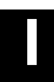
预设定

使用预设定功能可以在机床坐标系中重新定义系统的零点。在预设定中轴没有运动，它仅仅给当前轴的位置输入一个新的位置值。

驱动

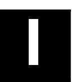
- SINUMERIK FM-NC 提供一个10V的模拟量接口，
  用于变频系统SIMODRIVE 611A。
- 数控系统SINUMERIK 840D
  通过一个快速数字并行总线与变频系统SIMODRIVE 611D相连。

■



D 索引

E 命令，标记



寄：

Siemens AG

A&D MC BMS
Postfach 3180

D-91050 Erlangen, Germany

Tel. +49 (0) 180 / 5050 – 222 [Hotline]

Fax +49 (0) 9131 / 98 – 2176 [Documentation]

E-Mail motioncontrol.docu@erlf.siemens.de

建议

更正

出版/手册：

SINUMERIK 840D/840Di/810D

编程说明－工作准备部分

一般文献

此信来自

姓名

公司地址

街道：________________________

邮编：__________ 地址: ______________

电话：__________ / ______________

传真：__________ / ______________

编程说明

订货号： 6FC5298-7AB10-3RP0

版本 03.04

当你阅读此刊物时若发现印刷错误，请以该表格通知我们。欢迎提出改进建议。

建议和/或更正：

Siemens AG
Automatisierungs- und Antriebstechnik
Motion Control Systems
Postfach 3180, D – 91050 Erlangen
Germany

www.siemens.com/motioncontrol

订货号: 6FC5298-7AB10-3RP0

在德国印刷